REQUIREMENTS DEFINITION WITHIN ACQUISITION AND ITS RELATIONSHIP TO POST-DEPLOYMENT SOFTWARE SUPPORT (PDSS)

by

M. Hamilton & S. Zeldin

November 1979

HOS HIGHER ORDER SOFTWARE, INC.

NOTICES

Contract P...
Vol. I, AD-A...

DISCLAIMERS

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government endorsement or approval of commercial products or services referenced herein.

DISPOSITION

Destroy this report when it is not longer needed. Do not return it to the originator.

HIGHER ORDER SOFTWARE, INC.
806 Massachusetts Avenue
Cambridge, Massachusetts 02139

(14) [illegible]

(1) Technical Report #22

Volume II

(15) DNACP[illegible]

(12) [illegible]

REQUIREMENTS DEFINITION WITHIN ACQUISITION

AND ITS RELATIONSHIP TO

POST-DEPLOYMENT SOFTWARE SUPPORT (PDSS)

APPENDICES

by

M. Hamilton & S. Zeldin

November 1979

Prepared for
United States Army Electronics Command
Fort Monmouth, New Jersey

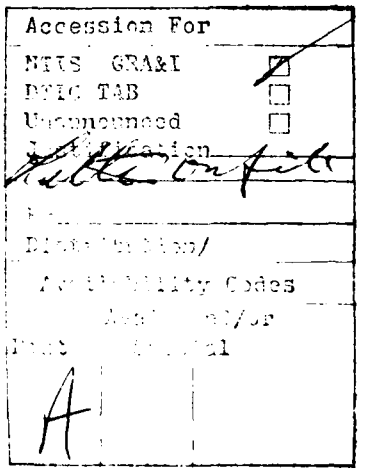
393027

APPENDIX I

ACQUISITION QUESTIONNAIRE

PART 1

SUMMARY OF RESPONSES[1,2]



[1]Footnotes refer to comments of respondents.

[2]Not all respondents responded to all questions.

TABLE OF CONTENTS

ACQUISITION OVERVIEW

QUESTION 1: What are the major problems encountered in system acquisitions you are familiar with? (Respondents where asked to check those that apply.)

ANSWERS: % of Responses	Problem Areas
100	Time slippage
91	Cost overruns
100	Failure of Service to fully specify what is required[1] [2]
73	Failure of Contractor(s) to meet specifications[2]
27	Failure of Service and Contractor(s) to deal realistically with new technologies[1] [2]
27	Justification of budget commitments
82	Competition for limited funds[1]
100	Changing requirements[1] [2]
91	User/developer interface[1] [2]
64	Implementation does not evolve from requirements[1] [2]
64	Reinventing the "wheel"[1]
18	Determination of how and when multiple contractor assignments should begin and end
64	Quality of management
55	Rapid management turnover rate
82	Lack of incentives to call a halt to any step of acquisition[2]
64	Clarification of knowns from unkowns[1]
64	No specific early warning signals for error detection available as management tools[1] [2]

[1] Importance in providing PDSS Planning and Implementation.

[2] Most probable result of developer and user having a poor communication interface.

QUESTION 2: Can you cite an example(s) of a lesson learned from a previous success or failure that was or is being used in the acquisition of a new systems?

ANSWERS: (Bullets indicate individual responses)

- Bureaucratic process interferred with learning process; competition (e.g., can not go to guys that can do a good job); too much guidance from Top; every program seems to be different.

- System must be fully tested in contractor's plant (laboratory) before being allowed to go into field tests. Systems integration testing has become recognized as a key factor before going into the field. Find-fix-test or "debug" in the field is costly and should be minimized. [author's note: see below for alternative opinion]

- Capable, competent people must be dedicated to at the rate of one (1) person/$250K per year. They must visit the contractor site, create tests, review progress and interact with the developer on at least a weekly basis. We have had fewer problems doing it this way.

- We never learn!

- Equal emphasis on software development as on hardware--lesson learned from TACFIRE.

- Test planning must parallel system development--lesson learned from TACFIRE. Used in Battery Computer System. Problem clarifying known-knowns, unknown-knows, known-unknowns, and unknown-unknowns.

- Lesson learned: Tactical Display System.

  Underestimating schedule/cost factors in development programs (e.g., TACFIRE/TOS/PLRS).

- Several with TOS. Based on lessons learned (primarily from TACFIRE) we are: (1) insuring properly sequenced deliverables with appropriate intermediate deliverables; (2) software contract properly structured; (3) provisions made for necessary visibility throughout development; and (4) overall program schedule modified away from "all success".

- One major lesson--continuity of management, and direction causing restart and redo. The Design-to-Price EW systems, aware of this problem experienced relative success by maintaining long term, high level management throughout the system design, development, T&E and acquisition.

- Success: Functional logistics analysts worked in teams with ADP analysts. Provided direct user/developer co-ordination; implemented established requirements; caused invalid requirements to be identified and withdrawn.

- Success: Find-Fix-Test mode provided rapid identification of problems, relative rapid resonse in fixing; and gave user much higher level of confidence in the system.

- Failure: No identification of Post Deployment SW Support. Continues to allow systems to be fielded and PDSS handled as expensive afterthought: incorrect documentation identified in CDRL's: no language guides; no upfront money to allow support personnel to learn with development of system.

- In general we seem to continue in our practice of the same old stupidities. Perhaps one small area of light is appear-ing in the specification process where we are beginning to specify software requirements as a separate entity from and on equal footing with hardware.

- The PATRIOT system is being developed and too late into the development cycle it was "Oh, by the way we did not get any guidance or requirements." Now the developer is trying to play catch-up. There is a serious shortage of personnel, funds, facilities and management.

ASSUMPTIONS

There are certain assumptions we all make as system managers or as system engineers. Respondents were asked to check the items below that they agreed with. They were also asked to list others that are not included in the items listed below.

ASSUMPTIONS: *Question 1 percentage of responses continued.*

% of Responses	Assumptions
64	Individual systems' needs are not in general unique[1] [2]
91	There are some very basic properties that are generic to all systems
45	Advanced technology more often evolves within an environment where systems are large and complex[3] [4]
64	Various disciplines are recognizing the importance of solving the problems of communication[5]
100	There is now an emphasis on "methodologies"[6] - hierarchical decomposition - front-end emphasis - integration of "modules" throughout a development process - requirements and specification languages
82	The power of a definition is not fully realized[7]
100	The power of simplification is not fully realized[8]
73	The power of commonality and abstraction is not fully realized[5]
55	The power of flexibility and extendability is not fully realized[5] [9] [10]
82	The difference between "good" modularity and "bad" modularity still needs to be better understood[5] [11] [12]
64	Algorithms and techniques are often misused interchangeably
91	Sometimes the same problems exist in the development of methodologies as exist in the problems the methodologies are intended to address[5]

---

[1]Management approach is general; technical properties are unique.
[2]Systems is rather broad. Prefer categorization of tactical systems since there are some basic differences between business batch processing versus tactical-real time systems.
[3]Apollo, yes; laser, no; general rule: reasonable technology evolves with money.
[4]Due to the large funds available to PM.
[5]Very important
[6]Not yet within Army as a whole.
[7]The lack of concise definitions cost dearly!
[8]Change "realized" to "explored".
[9]Humans are terrible at expressing requirements in advance.
[10]Change "power" to "necessity".
[11]Modularity itself needs better definition.
[12]Change "understood" to "explained".

ASSUMPTIONS: Question 1 percentage of responses continued.

% of Responses	Assumptions
82	Often in the attempt to compare methodologies there is the risk of comparing apples and oranges - techniques addressing very different problems - techniques intending to address the problem, but not effectively addressing it or not addressing it at all - techniques against no requirements or requirements not well defined - the "syntax" of methodologies instead of the "semantics" of methodologies - techniques that are rid of preconceived notions[13] with preconceived notions - techniques addressing the wrong problem - techniques with respect to completion or amount[13] of use rather than with respect to the problems they are solving
91	The behavior of large systems and their environment is being observed without the advantage of formal definition tools
55	Sometimes one set of methods is replaced by a new set of methods with only the solutions to problems of the old method as a consideration [5] [13] [14]
73	More work needs to be done in the area of knowing how to integrate methods and then integrating a system or systems
64	The choice of methodologies used can make the difference[13] between understanding a problem and not understanding a problem
64	The system problems we are attempting to solve are very[15] [16] [17] [18] basic ones, such as - how do people learn - how do people think - how do people communicate - how do people resource allocate

[13] Statement unclear.

[14] For example, we never incorporate new technology to do same function, but we always add requirements.

[15] At a certain level of abstraction

[16] Add common sense to list and add "why" for every "how".

[17] For example, a good manager has to be a leader but reverse is not true.

[18] These areas need much emphasis.

ASSUMPTIONS: Question 1 percentage of responses continued

% of Responses	Assumptions
64	The system problems we are attempting to solve are not unlike those in "older fields.[5] [19] [20]
45	The system problems we are attempting to solve provide much more visibility to some basic issues than related but more developed fields.[13]
36	A whole new set of basic principles is needed for developing systems.[21] [22]
27	More often than not, the user is the one who makes the decision as to what is technology and as to what is not. [23]

ANSWERS: Others (specify)

- Communication through a formal structure is almost non-existent across the multi-disciplines engaged in the development process.
- *Linking of requirements to specifications to sub-*systems to modules to operational criteria is not a clear hierarchy of steps.
- Architecture needs tools for requirement experimentation as well as the engineer needs tools.

---

[19]Qualify as "some" system problems.

[20]Change "we are attempting " to "we should be attempting".

[21]No, there are some meaningful principles needed to complement those currently existing.

[22]Change "basic" to "organized".

[23]Not in the Army.

UNDERSTANDING THE PROBLEM

Question 1: What types of data do you gather to understand better the environment of a particular user? (Respondents were asked to check those that applied.)

ANSWERS:	% of Respones	Types of Data
	64	Greatest complaints
	64	Wish lists
	64	Desirable properties already in the user's system
	73	Manual processes
	82	Interfaces[1]

ANSWERS: Others (specify)

- Operational tests (tests human factors reliability)
- Failures
- Definition and assignment of the threat
- System inputs/outputs (i.e., what the real function to be performed as opposed to the method currently trying to carry out the function).
- Requirements inherent in his mission
- User involvement in design decisions
- To the maximum extent possible I would ignore the how he does it today or thinks he wants it done. Include existing desirable properties where not deletereous (user acceptability).
- Operational doctrine.
- Chain of command - protocol.
- Response time requirements
- Implementation of manual processes into automated systems merely speeds up a perhaps inefficient operation. The system concept need be understood. The products desired stated. Then review the desired result against list above and validate or refute as appropriate.

[1]Very important.

QUESTION 2: What types of user support do you envision as desirable for your own environment? (Respondents asked to check those that applied.)

ANSWERS:

% of Responses	
36	Model of the user remains fixed and user performs same functions[1] - help speed-up functions with more standardization or automated tools which support his non-existent or already existing support aids - provide additonal information at certain checkpoints which help him make better decisions faster - prevent him from making errors, or recover from errors already made - provide best decision making information at the latest time possible (dynamic as opposed to static)
36	Model of user remains fixed but some of user's functions are replaced by automated aids
27	Change the model of the user
55	All of the above

QUESTION 3: If you could list five (5) of the most important lessons you have learned in developing systems, what would they be?

ANSWERS: (Bullets indicate individual answers.)

- a) Spent time formally defining the system requirements using scenarios
- b) Develop testing scenarios for the requirements and final system
- c) Simulate/simulate/simulate
- d) Get a completed tested design, build it, then test against b
- e) Give yourself plenty of time

[1] First model entire system.

- a) Pay now or pay more later. Skipping formalized incremental testing until "integration" is a disaster.
  b) Define specifically and completely what the functional performance envelope should be.
  c) Break-up functions into manageable groupings with inputs, outputs, processing and "testable" testing.
  d) START-STOP, GO-NOGO between phases, change or be able to change vendors. Include incentives.
  e) Continuous A and B level configuration management baselines (with ECP changes) are mandatory.

- a) Do your homework in advance -- plan, estimate, define requirements--most important, user must be intimately involved as system is being built throughout development process for his ownership and acceptability.
- b) Do not try to buy it cheap. "Cheap is cheap" - fully fund in advance in terms of time and money or do not start.
  c) Build in flexibility and GO-NOGO decision points and use them (implies good management and visibility). We need incremental checkpoints. To build a system and then go through DT-OT is too late. Then, only option left is to kill the whole thing. We wind up buying it anyway.
  d) Sometime or other have to stop changing requirements and freeze baseline.
  e) Treat software like hardware.

- a) Do a comprehensive analysis of the requirements.
  b) Put sub-requirements in priority order so as to address more important ones.
  c) Spend effort in front deciding which requirements you can and cannot address because of technology or money.
  d) Resist changing requirements - make sure user knows effects.

- a) Verbal and written communication of requirements in the language of the writer(s) is insufficient for disciplined action.
  b) Requirements vary over lead time contributing to longer lead time and life cycle.
  c) Schedule and dollars tend to be unrealistic as projected early in the development cycle.

d) Design reviews appear to suppress rather than expose development problems.

e) Development programs are not sufficiently flexible to take advantage of ensuing technology advances.

f) The final product cannot be sufficiently examined for its acceptability or unacceptability; therefore it is accepted.

- a) Unambiguous statement of requirements is most important consideration.

  b) Freeze the requirements early - postpone changes for later implementation.

  c) Plan for problems - don't expect all success.

  d) Need a good work breakdown structure to (a) grasp total problem, (b) obtain visibility for control.

  e) A few really good people are worth far more than dozens of "bodies." Corollary: The solution to major problems is <u>not</u> to throw more people in to work on them.

- a) Long term exposure to existing system - user actions - interfaces.

  b) Again, long term system concept formulation, with user interaction, with final specification.

  c) Solidify specifications and stick to it during the the shortest phased development.

  d) Demonstrate early and often at low level with simulations of expensive modules. After successful demonstration hide the system!!

  e) Program at the beginning of the project 10% contingency costs. Identify it as such. any program CUTS below the 10% level should cause reexamination of program-probability of success is now deminished!

- a) People do not understand "system."

  b) Automating manual procedures may not develp an efficient system.

  c) Failure to consider operational life and support requirements after development.

  d) Supertalent depart for new developments rather than remain for support and maintenance.

  e) Requirements must state "WHAT" and not "HOW".

UNDERSTANDING THE PROBLEM: Question 3 answers continued.

- a) The specification is inspecific.
  b) The user has no clear idea of his requirements; problem rarely understood, far less clearly articulated.
  c) The translation of requirements into a system specification often transcends reality.
  d) Specification detail when provided is often irrelevant.

- a) Clearly define requirements.
  b) Identify resources.
  c) Watch the contract. We (Army), in general, are poor contract writers/managers.
  d) Reinventing the wheel.
  e) Buying the same items numerous times.

QUESTION 4: What are the kinds of problems have you had in developing systems? (Respondents asked to check those that applied.)

ANSWERS: % of Responses	Problems
82	Problem not understood
82	Complex systems
82	Requirements always changing: new ideas or errors[1]
64	Not knowing how much detail is needed to understand a system definition[2]
100	Unrealistic estimates of computer time, manpower, calendar time, on-board computer space and time

[1]Not controlled.

[2]What constitutes satisfactory B-level specifications?

UNDERSTANDING THE PROBLEM[3]*: Question 4 percentages of responses continued.*

% of Responses	Problems
91	Poor visibility and traceability[4]
36	Asynchronous aspects of sotware and its interfaces
27	Class of problems include those which are time-critical and self-correcting (feedback)
73	Uncertainty of how much to test: Redundancies and omissions
55	Standards and disciplines not defined[5]
91	Ambiguous, implicit, too detailed, or incorrect requirements
73	Fragmentation of personnel
18	Unknown hardware effects on software[6]
64	Software must accommodate hardware
73	Management problems inherent in large systems: too little or too much[7]
91	Difficulty in measuring correctness of software[8] [9]
64	Software not transferable [10]
82	Symptoms rather than root problems treated
73	System not understood
55	No integrated goals [11]
64	No integrated methodology

[3]This list could go on and on ...

[4]And monitoring.

[5]Not used, even when defined.

[6]When in parallel.

[7]Too structured.

[8]And extent.

[9]Very important.

[10]But overcoming this.

[11]On paper - not in reality.

UNDERSTANDING THE PROBLEM: Question 4 percentages of responses continued.

% of Responses	Problems
55	Structure of system development process not flexible enough to encourage multiple technologies, vendors, competitive innovation and multiple sourcing[12] [13] [14]
91	Unrealistic schedules[9]
82	Limited resources[9]
55	Difficulty in measuring effectiveness of software methodology/tools
55	Costly and lengthy efforts
82	Lack of sufficient documentation: too little or to much[15] [16]
91	The problems of Parkinson's law
91	Poor communciation
55	Communication lags
73	Communication interfaces not defined
36	Lurking errors
27	"Man-rated"[17]
45	Cannot be verified in the real world[18]
73	System does not live up to expectations
73	A need for automatic error detection schemes
55	Lack of flexible reconfiguration schemes during development and real time
55	Misunderstandings about capabilities of support systems

[12] Separate this list.

[13] Government policy precludes.

[14] Dollar limited, not structure.

[15] Vague, need intent of completeness.

[16] Too much.

[17] Unclear statement [Author's note: This means a system in which an error could cause the death of a human.]

[18] Not biggest thing in Army.

UNDERSTANDING THE PROBLEM: Question 4 percentages of responses continued.

% of Responses	Problem
73	Design by auditorium[9] [19] [20]
45	Paradox of redundancy management schemes
82	Improper structuring of incentives for contractors and Government management personnel[9]
73	Specific and narrow scoped interests[21]
45	Difficulties in personnel attitudes toward cooperation[9] [22]
82	False economies[9] [23]
55	Over sophistication
55	Creation of "urgent" problems by failure to anticipate troubles or respond expeditiously[9]

ANSWERS: Others (specify)

- Reliance on contractor to tell Government what is needed when he's got it and how great it is
- Improper testing-test to specifications vs. error testing
- Lack of understanding of what software is
- Lack of flexibility for technology insertion of minimum risk to schedule and cost. Sunk cost often precludes significant performance enhancements.
- Design reviews that do not reveal design problems
- lack of qualified project management personnel
- "Money-rated"
- Parkinson was an optimist
- We still try to justify systems in terms of numbers of personnel saved or eliminated

---

[19] Specification by committee

[20] Requirements by committee

[21] Lack of system perspective

[22] Not invented-here syndrome!

[23] Regulations prevent Army from buying enough-other side of Parkinson's law.

QUESTIONS 5: Pick a system or a target for development. What types of requirements affect its development[1]? (Respondents were asked to check those that applied and to add other items in areas where applicable[2] [3].)

ANSWERS: % of Responses	System/Target
45	Customer[4]
73	Mission
82	User
55	General tool requirements
64	Methodology
64	Host facility[5]
82	Support systems[6]
	- within target system
	- within development tools needed/ desired to develop system
	- requirements imposed by tools already developed
55	Pre-design system execution requirements[7]
73	Standards
82	Operational reliability
100	Testing[8]
45	Statistical gathering[9]

[1] Seems to be a misnomer in some instances.

[2] Customer, mission and user requirements inseparable.

[3] User = Customer in Army; mission defined by user.

[4] PM is customer in Army [author's note: see Note 3 for alternate opinion].

[5] Least important.

[6] Add "within host system".

[7] Unclear statement. Does this mean simulation or modelling?

[8] Repeatable and recursive.

[9] This is generated for testing.

UNDERSTANDING THE PROBLEM: Question 5 percentages of responses continued.

% of Responses	Problem
55	Overall goals[10] [11]
ANSWERS:	Other (specify)

- Funding (fiscal)
- Lack of central responsible activity
- networking requirement because of its potential complexity and influence of design
- Management structure (i.e., the DoD acquisition system

QUESTION 6: What do you consider the root problems to be in developing a system? (Respondents asked to check those that applied.)

ANSWERS: % of Responses	Root Problems
91	Understanding a system and its environment[12]
64	Communication within and between systems[14]
55	Resource allocation within and between systems
64	Flexibility, both in development, and in real time
	Other (specify)

- Not understanding the problem to be solved plus people doing what they want to do and not what they ought to do equals disaster
- Algorithm development
- System concept (i.e., centralized/decentralized)
- Scaling of subsystems in accordance with resource allocation
- Total life cycle
- Management ignorance of the difference between hardware and software

---

[10] Performance bounds.

[11] Goals and objectives, as opposed to requirements are the first things discarded when the going gets tough: they are front-end window dressing.

[12] Very important.

[13] Plus system elements.

[14] Not as important as others

CHOOSING AN EFFECTIVE METHODOLOGY

QUESTION 1: Without taking the advantage of formal definition tools, we are not taking advantage of the power of several side benefits. (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Formal Definition Tools
82	Definition[1]
91	Simplification
82	Commonality
45	Abstraction
73	Flexibility
73	Extendability
64	Understanding what the system is that is being defined[2]

ANSWERS: Others (specify)

- Readability by reviewers and users representing various disciplines.
- Trackability - can one follow functional processes across OS-applications programs - common processes.
- Standards (documentation, conventions, data element dictionaries, protocol)

QUESTION 2: Which viewpoints of a system are you most concerned with? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Viewpoints
45	Object
9	Name
82	Definition
64	Description
82	Implementation[3]
45	Execution
	Other Verification and Validation

---

[1]Please define.

[2]Appears redundant to 1 and 2.

[3]What is the difference between implementation and execution? [Author's note: Implementation prepares for execution, see Section 4.0 of Volume 1.]

QUESTION 3: Listed below are a series of questions and answers. (Respondents were asked to check those answers that they agreed with.)

ANSWERS: % of Responses		Questions and Answers
82	*Question:*	How can we tell if a methodology will work better than no methodology at all?
	Answer:	Compare the properties of the methodology with those used in an existing development with respect to a well defined set of requirements for consistency and completeness.
64	*Question:*	How do we choose between one methodology and another methodology?
	Answer:	Compare the properties of the two methodologies with respect to a well defined set of requirements for consistency and completeness.
73	*Question:*	What is the difference between using a methodology and using "smart" people?
	Answer:	The smartest person, by definition, would apply an effective methodology. An effective methodology would far exceed the advantages of a smart person applying his techniques in an ad hoc manner, since all the intricacies of a complex system are by its nature beyond the grasp of one human being. The designs of all smart people must be integrated[4,5]
55	*Question:*	How do we use a methodology without impacting deliverables of an on-going project.
	Answer:	Choose those aspects of the methodology which find errors or which expedite the design and implementation process[6].

[4]With those of dumb people.

[5]But please find me a good smart person rather than a methodology because of the added flexibility. But since smartes are not available...

[6]Might agree if the answer was couched in a different language. This is a short-sighted approach.

CHOOSING AN EFFECTIVE METHODOLOGY: Question 3 percentages of responses continued.

% of Responses		Questions and Answers
73	*Question:*	How do we convince management, designers, and users to use different approaches?
	Answer:	A different methodology should be demonstrated within the environment of the people who will use it[7] [8].
100	*Question:*	What creativity is left for the engineers if a methodology has constraints?
	Answer:	An effective methodology should support creative designs and not constrain them from producing better designs but rather constrain them from producing errors[9].

QUESTION 4: What methodologies have you heard of before? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Methodologies
36	HDM
73	SREM
55	PSL/PSA
100	Structured Design
9	Warnier
100	HIPO[10]
36	Jackson
36	SADT
55	Software Factory
82	HOS

[7]Success stories heard from peers is most effective. It is getting the initial implementation that is hard.

[8]And it must be effective.

[9]Smart assistant!

[10]Not really a methodology, just a tool.

CHOOSING AN EFFECTIVE METHODOLOGY: Question 4 percentages of responses cont.

% of Responses	Methodologies
Answers:	Others (Specify)
9	• CARA
18	• IORL
9	• SVD
18	• Threads
9	• Chief Programmer Teams
9	• SAM
9	• SCALD

QUESTION 5: Which of these are you most familiar with?

% of Responses	Methodologies
45	Structured Design
45	HOS
36	HIPO
9	IORL
9	SVD/Threads
27	SREM
18	PSL/PSA
9	HDM
9	SCALD
9	About the same of each

QUESTION 6: Would you recommend any of these methodologies?

% of Responses	Methodologies
9	Structured Design
18	HOS
9	HIPO
9	IORL
9	SVD
9	SREM
9	PSL/PSA

CHOOSING AN EFFECTIVE METHODOLOGY: Question 6 percentages of responses cont.

% of Responses	Methodologies
9	HDM
9	Knowledge not deep enough to comment
18	Most of them over none at all

QUESTION 7: What properties do you consider to be desirable ones for a system development methodology? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Desirable Properties
91	Techniques for defining systems which are consistent and logically complete.
73	Techniques which are within themselves consistent and logically complete, both with respect to each other and to the system to which they are being applied[11].
82	A standard set of definitions which reside in a well publicized and evolving glossary[12].
91	Mechanisms to define all of the relationships which exist in a system environment[13].
64	Mechanisms to define all of the relationships which exist between possible viewpoints (or development layers) of a system[14].
73	Mechanisms to consistently and completely define an object and its relationships formally.
91	Provisions for modularity of the right kind and prevention of separation of the wrong kind.
55	Provisions for a set of primitive standard mechanisms which are used both for defining and verifying a system in the form of a hierarchy.

[11]Do you mean "applicability?" [Author's note: yes.]

[12]That is, a well defined methodology.

[13]Applicability

[14]Change "all" to "of the relevant"

CHOOSING AN EFFECTIVE METHODOLOGY: Question 7 percentages of responses cont.

% of Responses	Desirable Properties
73	Provision for an evolving set of more powerful (with respect to simplicity and abstraction) mechanisms based on the standard set of primitive mechanisms.[15]
100	Allow system engineers to communicate in a language, with common semantic primitives and a familiar dialect, which is extensible, flexible, and serves as a "library" of common data and structure mechanisms.
82	Provisions for a development model, including a set of definitions, tools, and techniques, which effectively support a given system development process.[16] [17]
64	Not only must a methodology be effective, but it must also be able to be used as well, and the results of that use should be made available to others.[16] [18]
64	There should be an explicit set of rules that must be followed in order to proceed from one level, or one layer, to another so that - one is able to determine if a function has been properly decomposed with respect to complete replacement; no more, no less - one is able to specify a system without data conflicts or timing conflicts
45	Each node on a given hierarchy integrates all aspects of control, architecture, and viewpoints.
82	On a given hierarchy one knows when a system definition is complete.[16]

[15]Not necessary for a given system, but desirable.

[16]Motherhood.

[17]Need flexibility to tailor it.

[18]Here is where we lose effectiveness in the application of technically sophisticated or not-so sophisticated methodologies. We must be willing to apply people to do the job and often we are unwilling to do so in the process of development.

CHOOSING AN EFFECTIVE METHODOLOGY: Question 7 percentages of responses cont.

% of Responses	Desirable Properties
	Answers: Others (specify)
	• The techniques should be easy to understand and use by the system designer.
	• Technique should be interpreted by cognizant but not super-sophisticated personnel.
	• As a language, technique should be of the order of current HOLs.
	• Methodology should affort traceability or linking to lower level specs/programming languages.

REQUIREMENTS FOR FORMAL METHODS

QUESTION 1: What kinds of ambiguity have you had experience with in your system development processes? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Kinds of Ambiguity
64	Definition of terms
82	Definition of specifications/ requirements
	Answers: Other (specify)
9	• Hierarchy of functional processes or priorities in processing.
9	• Lack of testability of components and functions.

QUESTION 2: Which terms have you had difficulty with? (Dashes indicate individual responses.)

ANSWERS:
- Software
- Modular
- Real-time
- Standardization
- Interoperability
- Embedded
- Protocol

REQUIREMENTS FOR FORMAL METHODS: Question 2 answers continued,

- Priority
- Security
- Continuity of operations
- Specifications
- What the system is to do (lack of specifically of functional allocations)
- Requirements
- A, B, C Level Specs
- Design
- Test
- Evaluation
- Quality Assurance
- Production Engineering
- System Definition
- Reliability, Availability, Maintainability (RAM)
- Verification and Validation[1]

QUESTION 3: What types of units should be defined in a system definition? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Types of Units
91	Data types
91	Functions
82	Structures (relationships between functions)
Answers:	Others (specify)
9	• limiting constraints

[1]Definiton of one respondent: Verification is levels and layers, validation is execution, that is, does what it is supposed to do.

QUESTION 4: Do you check for inconsistent definitions? How?

ANSWERS: % of Responses	Response
54	Yes
9	No
9	Like to think so
9	Not very well
9	No response

HOW: % of Responses	Method
27	Eyeball or manual inspection
9	Indirectly through review process .
9	Cross referencing and cross checking .
9	Comparison

QUESTION 5: Do you check for incomplete definitons? How?

ANSWERS: % of Responses	Response
73	Yes
9	No
9	Not very well
9	No response

HOW: % of Responses	Method
36	Eyeball or manual inspection
9	Referring to basic documentation or originator.
9	Specification writing and review

QUESTION 6: Do you check for redundant defintions? How?

ANSWERS:	% of Responses	Response
	45	Yes
	18	No
	9	Not really

HOW:	% of Responses	Method
		Eyeball
		Manually[2]
		Multiple names for common items across modules/functions

QUESTION 7: Do you use any standards or methods to encourage modularity[3]?

ANSWERS:	% of Responses	Response
	36	Yes
	27	No

Do these methods always help, if no, are some types of modularity methods error prone?

ANSWERS:	% of Responses	Response
	9	Yes
	9	No
	10	Sometimes

ANSWER: Nine percent of the respondents felt that some types of modularity methods are error prone. 91% did not respond

---

[2]And this is time consuming.

[3]Aren't you really talking about composition methods? [Author's note: yes]

REQUIREMENTS FOR FORMAL METHODS: Question 7 answers continued.

If yes, how do these methods help? (Bullets indicate individual answers.)

- Surface inconsistencies and redundancies.
- Allowing some level of reconfiguration to satisfy changing requirements - assists in phased development
- Need management controls to limit exceeding modularity constraints

SYSTEM VIEWPOINTS

QUESTION 1: What definition techniques do you use? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Definition Techniques
64	Hierarchical decomposition
9	Algebraic data types
Answer:	Other (explain)

- Only the process afforded by MIL-STD 490-A,B,C Level Specs

QUESTION 2: If you use a hierarchical decomposition technique, which one do you use?

ANSWERS: (Bullets indicate individual answers.)

- HIPO
- HDM
- Top Down Structured Analysis/Design
- Informal Decomposition

QUESTION 3: What description techniques do you use for system requirements or specifications[1]? (Respondents were asked to check those that applied and to indicate which ones applied.)

ANSWERS: % of Responses	Description Techniques	Which ones?
18	Specification/Requirement Language	PSL/PSA
9	Program Design Language	PDL
45	Higher Order Languages	CMS-2, SPL-1, ADA (1981), TACPOL, IFTRAN, PASCAL, FORTRAN, COBOL
73	English	
36	Graphics (diagrams)	Digital System Diagram
9	Others	MIL-STD 490
9	None	

QUESTION 4: What implementation techniques do you use? (Respondents were asked to check those that applied and which ones applied.)

ANSWERS: % of Responses	Implementation Techniques	Which ones?
55	Compiler driven	TACPOL, FORTRAN, HAL/S
18	Analyzer	DAVE/PET
36	Manual	
9	Others	MIL-STD 490

QUESTION 5: What execution techniques do you use? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Execution Techniques
55	Simulation
18	Hybrid
36	Static automatic

[1]Army does not dictate; one exception is ATLAS.

SYSTEM VIEWPOINTS: Question 5 percentages of responses continued.

% of Responses	Execution Techniques
27	Static manual
36	Dynamic automatic
18	Dynamic manual
9	Modelling
27	Interface
45	Performance
9	Off-nominal Stress Tests

QUESTION 6: When do you think it will be possible to go directly from a requirement/specification to execution code? (Respondents were asked to check projected time frames.)

ANSWERS: % of Responses	Time Frames
18	One (1) year[2]
9	Five (5) years
0	Ten (10) years
18	Over ten (10) years
18	Never[3]

THE SYSTEM DEVELOPMENT PROCESS

QUESTION 1: What components do you think should go into a model[4,5]? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Components
55	Goals
64	Methodology

[2]Possible in one year, practical in five, wide-spread use in ten.

[3]You must design the system.

[4]Unclear statement: Also do you mean system model, development process model? [Author's note: System development process.)

[5]This is more difficult than 6 above to obtain.

% of Responses	Components
55	Definitions
45	Library
55	Disciplines and their checklists (Management, design[6], implementation, verification, documentation)
55	Phases and their checklists
55	Relationships between and within phases and disciplines
45	Tools and techniques
36	Personnel structure
Answers:	Other (Specifiy)

- Standard operating procedures
- Restrictions
- System inputs/functional characteristics, outputs, system interfaces and communication paths
- Testing

QUESTION 2: We make the assumption that the disciplines listed below correspond directly to the system viewpoints listed beside them. Do you agree?[6] [8]

Disciplines	System Viewpoints
Management	Control
Documentation	Description
Design	Definition
Resource Allocation	Implementation
Verification[7]	Execution[7]
Others	

[6]In general yes, but they overlap.

[7]All except this one.

[8]e.g., resource allocation ambiguous. Part of the management process on one hand, may pertain to internal system resources also.

SYSTEM DEVELOPMENT PROCESS: The answer to Question 2.

ANSWERS:	% of Responses	Response
	27	Yes
	9	No
	64	No response

- Would prefer:

Discipline	Viewpoint
resource allocation	segmentation
execution	assignment or modularization
verification	implementation satisfies requirement

QUESTION 3: We believe that an ultimate goal for a system development manager is to replace himself by automation. Do you agree? (Author asked for explanations.)

ANSWERS:	% of Responses	Response
	18	Yes
	55	No
	27	No response

Explanations: (Bullets indicate individual responses.)

- Yes, tired of working.
- No, it is to work himself out of a job.
- Yes, for requirements, provided tools are available, the resultant system should do precisely everything that was intended.
- No, there are elements of management which are human and not quantifiable
- No, have to insure that requirements are really being met.

- Not really. The system development managers must set the stage for the development effort, determine the means for measuring progress, and resolve the human failings and misunderstandings as they arise. The machine cannot carry out these functions as well.
- No, this is like saying that a command and control system should issue commands and control their execution. That is what a control system does on a micro level. At the macro level, the commander uses a C&C system to assist him in his making decisions, and then assists him in seeing that these command decisions are communicated and implemented. The manager should never be out of the loop either.
  [Author's comment: See Section 4, Volume 1, for classification of leader vs. manager.]

QUESTION 4: Can you envision a way in which more formal, modular, or communicable approaches could help you as a manager? (Bullets indicate individual responses.)

ANSWERS:

% of Responses	Response
55	Yes
45	No response

- Eliminate layers of management
- If requirements are set forth completely, then implementation, testing, acceptance and follow-on modification and PD can be controlled all with minimum amount of resources.
- The most common means of conveying ideas, instructions, relationships, and meanings is the English language and it is largely insufficient in spoken or written form for system development programs involving people schooled in a variety of disciplines.
- Yes, bookkeeping. Checking of design decisions, maintaining and checking descriptions for completeness and consistency.

- Provide a more accurate picture of indicators which will allow more rapid management action. This allows better use of resources and aids in "on time, within cost".
- Yes. (Well, you did not ask for a description of "how".)

QUESTION 5: What are the overall goals in developing your system? (Respondents were asked to number in order of importance.)

ANSWERS: Nine percent of the participants did not respond.

OVERAL GOALS	RESPONSES									
Satisfy the customer	1	1	1	1	1	1	1	1	1	1
Satisfy yourself, technically	3	3	2	4	2	2		1	2	1
Prepare for follow-on work	2	2	3	3	3			2	4	
Other (explain)										
• Help my company make profitable				2						
• Make a contribution[8]						3				
• Satisfy the bureaucracy[9]							1			
• Satisfy yourself, professionally									3	

[8]That is, improve the technology, company, Army, Country, etc. It is possible to please the customer, satisfy yourself, and, yet, contribute nothing, because the task itself was not worthwhile.

[9]This requires some constraints on satisfaction of the customer.

QUESTION 6: Below is an example of one way to produce a system design. Does this method conform in concept to your own? (Respondents were asked for comments.)

Jot - jot down notes about functions that are in the target system; organize and recorganize the notes until they exist in a hierarchical form to work with.

Plan - complete as much as possible a commented hierarchy with questions which reflect specification properties of target system functions. Use object lists, library, standards as aid in asking and answering questions.

Interview - use the definition to interview system, support system engineers and customer to fill in the missing parts. Enter your own original information acquired from questions on standard forms into the requirements data base.

Produce system - define new specification mechanisms and incorporate into system definition.

Translate - analyze statements for interface consistency (proper decomposition). Check to see if problem is defined as originally intended.

Approve - go through approval channel for acceptance. If not finished, redo the process.

ANSWERS:

% of Responses	Response
55	Yes
18	No
27	No response

COMMENTS: (Bullets indicate individual responses.)

- Yes, but Government lags in interviews.
- Yes, the specs must reach a complete and accurate state and then become baselined and controlled by the customer. Otherwise contractor will shift the "spec" to accomodate the "as built" system which will undoubtably not satisfy user requirements, thereby entailing a redesign and costly process to provide what the customer really wants. Make a commitment and then build it.

SYSTEM DEVELOPMENT PROCESS: Question 6 comments continued.

- No, I use simulation at a much earlier level. I try to identify the hard parts technically difficult problems and solve them easily even though they may be at a lower level.
- This appears to be a top-down, cooperative user/developer scheme to arrive at a mutually acceptable design description. Hopefully, there is more interchange at the "jot" stage than appears here. It must not become an independent effort on the part of the designer and he must not proceed so far along as to confound the user.
- Not really. May apply to micro level where a great deal of freedom exists and task definition leaves room for modification of task (and commensurate modification of related tasks). At macro level, this freedom does not generally exist. Customer predetermines much of the information you are looking for above.
- Yes, unfortunately!
- Yes. One item not identified above I consider extremely important prior to starting the system design, but included in my concept of design is "study the big picture". Comprehend where your system fails in the "total system" and understand the functional area of application. I have found this approach most effective as it provides a reference point for the designer to talk to the user in the users frame of reference and not the technical frame of reference of the designer.

QUESTION 7: What techniques do you use to make the operation of your facility more efficient? (Respondents were asked to check those that applied.)

ANSWERS:

% of Responses	Techniques
18	Smart snapshots (memory and timing)
45	Detail of simulation modelling

SYSTEM DEVELOPMENT PROCESS: Question 7 percentages of responses continued.

% of Responses	Techniques
27	Number and detail of user aid requests (e.g., trace, clock pilot, print-out options)
18	The simulation should verify only the module necessary to satisfy the objective
64	Use real module for performance testing
27	Use "fast" module where logical verification only is required
18	Order of testing a module where logical verification only is required
45	Use right tools for the test objective (e.g., analyzer to statically verify possible memory conflicts, possible priority conflicts, timing, error recovery, etc.)
27	When submitting new runs - don't run variants until it is assured that one deck has run through successfully[10]
55	Use support software which monitors computer usage.

Thirty-six percent of the participants did not respond.

QUESTION 8: Which tools and phases apply to your target system development? (Respondents were asked to check those that applied.)

ANSWERS: (The percentages are tabulated in the chart that follows.)

Fifty-five percent of the participants did not respond to Question 8.

---

[10] Important

SYSTEM DEVELOPMENT PROCESS: Question 8 percentages of responses.

	CONCEPT FORMULATION	PROGRAM VALIDATION	FULL-SCALE DEVELOPMENT	PRODUCTION AND DEPLOYMENT	OTHER: DOCUMENTATION	T&E
COMPONENT TOOLS		9				
- Requirements/Specification Language	36	27	18	9	9	9
- Analyzer	9	18	45	45		9
- Static Resource Allocation Tool		9	9	9		9
- Other						
SUPPORT TOOLS		18	27	27		9
- Data-base Structure	18	18	27	27		9
- Resource Monitoring			27	18		9
- Inter-revision Updater		9	18	18		9
- Collector			9	9		9
- Text Editor	9	9	27	27		9
- Text Formatter	9	18	18	18		9
- Simulator	18	18	9	18		9
- Emulator	9	18	18	9		9
- Performance Monitor		9	27	9		9
- Other						
INCREMENTAL TOOLS		18	18	18		
- Assembly Language		9	36	27		9
- Macro-processor/ Assembler			18	18		9
- Higher Order Language		9	27	18		9
- Compilers		9	27	27		9
- Structured Flowcharter	9		27	27		9
- Interactive Debugger			27	18		9
- Interpreter			27	18		9
- Other						

QUESTION 9: Could some tools be the same ones that are not the same today?[11]

% of Responses	Response
9	Yes
91	No Response

QUESTION 10: The following is a list of recommendations we made for developing a system. (Respondents were asked to check those that they believed would be beneficial.)

ANSWERS: % of Responses	Recommendations
82	Define systems at front-end with hierarchical definitions (integrate from the beginning)
82	Perform design and verification role on all phases making use of a formal definition technique[12]
82	Provide interface specification document - common definition of terms dictionary - common data dictionary - common structure dictionary - common function dictionary
64	Provide user manual which provides checklists and explains - how to interpret interface specification document (for users) - how to design modules for adding to the "library" of the interface specification document (for designers) - how to define standards for system development (for managers)

[11]Unclear statement: Do you mean that an improved tool replaces a less effective one? Or do you mean that a tool can perform in multiple phases, that is, program validation/production?

[*AUTHOR'S NOTE:* The latter.]

[12]Very important.

SYSTEM DEVELOPMENT PROCESS: Question 10 percentages of responses continued.

% of Responses	Recommendations
55	Provide guide to implementation - how to go from specification to computer - how to provide for reconfiguration of functions in real time
	Provide a development model[12]

Answers: Others (specify)

- Suggest alternative organizational structures. The process will be no more effective than the array of personnel assigned and trained to carry it out. The finest development process conceivable is worthless without competent people to carry it out.
- "How to" manual for the managers of the requirements determination organizations.

QUESTION 11: We believe that to change to techniques which will make a positive impact in system development, that an initial investment is necessary to define and develop a general model for defining systems. Would you be interested in revamping your own methods[13] [14] ?

ANSWERS:

% of Responses	
64	Evolutionary[15]
18	Revolutionary[16]
0	Not Certain
36	No Response

[13]Do not have my own methods.

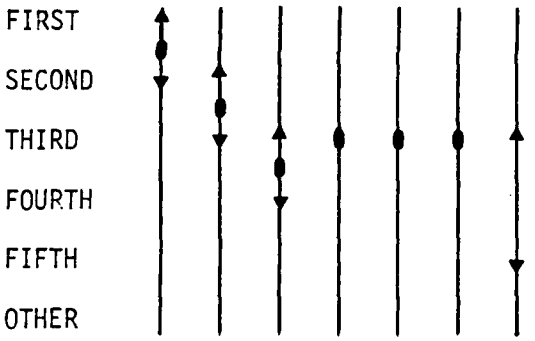
[14]Evolutionary in the course of an on-going program; revolutionary at the outset of a next project with the qualification that the methods have proved valid.

[15]With constraints.

[16]How?

QUESTION 12: On the next page is a table illustrating the evolutionary nature of software, as we see it. Where do you think your own development techniques fit in?

ANSWERS: (The bulleted item is an individual response.)



- Third: there is yet to be one example of a specification language applied in the development of our Army tactical systems of which I'm aware. We do accept the fact that the discipline afforded, avoidance of semantic problems, misconceptions, reduction in ambiguities, and incompleteness, as well as gains in testability would result from the use of a formal specification language; perhaps one that would be a close derivative of a requirements language. Unfortunately the step has yet to be taken and we certainly won't get to the "fifth" unless we move to the "fourth".

GENERAL MANAGEMENT QUESTIONS

QUESTION 1: Select a project of reasonable size which you managed. Suppose you were given a chance to do it over again any way you wished. What would you do the same way? What would you do differently?

ANSWERS: (The bulleted items on the following page are individual responses by the participants.)

	1st Generation	2nd Generation	3rd Generation	4th Generation	5th Generation
FOCUS OF APPROACH	TALK TO COMPUTER	MAKE IT EASIER FOR PROGRAMMER TO TALK TO COMPUTER	CONCENTRATE ON RELIABILITY AND COST SAVINGS IN PROGRAMMING PROCESS	CONCENTRATE ON RELIABILITY AND COST SAVINGS IN FRONT-END DESIGN	HARDWARE/SOFTWARE/FIRMWARE BECOME INTERCHANGEABLE AT A MODULAR LEVEL
DUAL RELATIONS	ASSEMBLER ↕ ASSEMBLY LANGUAGE	COMPILER ↕ HIGH ORDER LANGUAGE (HOL)	PREPROCESSORS AND "STRUCTURED" COMPILERS ↕ STRUCTURED PROGRAMMING	ANALYZER ↕ REQUIREMENTS LANGUAGES	COLLECTOR ↕ MODULES DERIVED REQUIREMENTS LANGUAGES
TIME FRAME	1950's TRADITIONAL APPROACH	1960's	EARLY 1970's	LATE 1970's	1980's

FIGURE A1-1 THE EVOLUTIONARY NATURE OF SOFTWARE

GENERAL MANAGEMENT QUESTIONS: Question 1 responses.

- Would maintain hard line with requirements developers. Would use more innovative, state-of-the-art software techniques.
- Same: Modularity, a la Parnes, of system functions into a set of "minimum essential functions" for phased development.

  Different: Closer tie with contractor to input "governement view" on design options selected during development.
- Haven't really managed one to completion yet.
- Keep competent people. Would want more concentration on front-end planning and time to test the concept through simulation. More time to organize the program before it becomes final. More and better cross communication among participants. More visibility for people to see the system take shape beyond their own effort.
- Have a non-computer specialist (management, finance, etc.) act as co-project engineer.
- Firmer requirements/more simulation and closer contract monitoring/testing. Use HOL/improve development and test environment.
- Coding would be the absolute last thing to be done. Descriptive flows would be complete and all "loose" ends tied off and squared away. Structured walk throughs. Establish reasonable B-level descriptions and maintain CM baseline.
- Not in the way I managed it, but in the way it was managed.
- Very little in the same way. Would start with better definition of requirements. Started with poor A-level specifications. kept changing as system grew. Nature of R & D is 2 steps up and one back and so in some sense wasteful. Would verify, change acquisition process to have prime bidder demonstration and auditor for software. Biggest thing, don't try to buy it cheap. In both cases the Government allows contractors to "buy in" (i.e., a contractor estimate less than government estimate should be suspected).

QUESTION 2: What criteria do you use to know what constitutes a phase of development? (For example, how do you know if a phase is complete?)

ANSWERS: (The bulleted items on the following page are individual responses by the participants.)

GENERAL MANAGEMENT QUESTIONS: Question 2 responses.

- Primarily use testing against specific criteria on definable products (i.e., modules, programs, subsets, system).
- Minimum essential set of functions. Time/Cost, Demonstratable performance.
- Milestone accomplishment. Predefine what tasks are to be completed during a phase along with criteria to determine "completion" (satisfactorily).
- We are prone to carefully set key events or milestones signifying an interim completion point (i.e., initial test of edit program). Where we stumble is how do we validate the test, how do we know it was successful enough. Conclusion, we are sloppy on our metrics. I try to use some objective measure.
- We define deliverables and review them when complete.
- If the performances and/or functional requirements have been satisfied, then the activity is complete for the overall system.

QUESTION 3: What are the most important characteristics you look for in the people you hire? (Some respondents identified characteristics in order of importance.)

ANSWERS: % of Responses	Characteristics	Responses by Importance		
64	Intelligent	2	4	2
64	Motivated	1	2	1
18	Educational background[1] [2]	5	5	
0	Attractive personality	3	3	
64	Experience	3	1	1
Answers:	Other (explain)			

- Ability to work in tribe environment.
- Character (Number 1), i.e. reliability, honesty, dedication, etc. An intelligent, motivated individual gains experience and learns quickly. I'm becoming less impressed with an individuals experience.
- Doer, gets the job done with independence.

---

[1]Depending on position - intelligent and motivated may be more trainable and productive than an individual with educational background.

[2]Most people in this business have good credentials. More important at the beginning of a career, then other factors begin to take on more significance.

QUESTION 4: Are you able to envision your own system development process as being systematic?

ANSWERS:	% of Responses	Response
	55	Yes[3] [4]
	9	No
	36	No Response

QUESTION 5: What tasks in your project have been converted from ad hoc methods to standardized methods? Which have been automated?

ANSWERS: (Bulleted items are individual responses by the participants.)

- None
- Standardized documentation; systems analysis documentation, test packages. None automated.
- Modularity of functions, standardized software documentation, standardized languages/support software.
- Limited standardization - specification testing.
- Limited automated-testing
- Improvement in use of higher order languages, test techniques. We are better at handling the period from the midpoint of the project to the end than from the beginning to the midpoint.
- Requirements/Design - formal. Configuration control - automatic.
- A,B,C levels and testing. Automatic verification system tool DoD 5000.29, 5000.31s.xx, DoDI.

QUESTION 6: In what ways have you and your people advanced the state-of-the-art in various technology areas?

ANSWERS: (Bulleted items are individual responses by participants.)

- Parnas technology, HDM, PSL/PSA, HOS (a little), SDL, Software Engineering, standard hardware/interfaces, protocols.
- Hardly any.

---

[3] If left alone!

[4] Hopefully. Do you mean my contractor's or what we specified? [Author's note: the latter.]

- Good at determining a better way to carry out development programs but poor at putting it into practice.
- Code generation, software verification and test tool development, software management.
- Awareness. HOS contract, CENTACS program.

QUESTION 7: What would you like to do next with respect to technology advancement?

ANSWERS: (Bulleted items are individual responses by the participants.)

- Develop a means for assessing software reliability.
- Standardized architecture with certification techniques to allow the purchase of "systems" vs. computers.
- The variety of software tools available boggle the mind. Would like to be involved in an effort to integrate a few into an orderly, logical process applicable to a real-world system development.
- Improve integration of requirements and design.
- Technology insertion via MCF, DoD-1 convergence, Production of OS, support software tools.

QUESTION 8: What properties do you want your system to have? (Respondents were asked to number in order of importance.)

ANSWERS:

Properties	Responses by Importance						
Modular[5]	3	6	1	4	5		2
Reliable	2	1	1	2	1	1	1
Efficient	1	4		5	6	1	7
Easy to understand[6] [7]	6	5	1		3	1	5
Traceable[7]	8	3			2	1	6

[5]Under broader category of extendable.

[6]Use ninth-grade maintenance and user.

[7]These should be inherent, cannot rate with others.

Properties	Responses by Importance						
Transportable[7]	7	2	1		8	1	8
Interoperability[8 9]	4	8	1		7		4
Flexible[5]	5	7	1	4	4	1	3
Other (explain)							
• Correct[10]				1			
• Maintainable				3			

QUESTION 9: a) What are your greatest costs? b) Do you know how to reduce them? c) Do you want to reduce them?

ANSWERS: (Bullets indicate individual responses by the participants.)

A

- Correct implementation, T&E, verification.
- People
- People over long periods of time.
- Personnel
- Software people; labor intensive but lousy productivity.

B

- Support enhancement.
- No, other than improving their productivity.
- Need to reduce time factor. Need shorter development process. Can do this by increasing productivity of people. We must get a better product earlier.
- More automation.
- Bring about discipline as per hardware process.

C

- Of course.
- Yes

[8] Difficult question: specific system considered does not have interoperability reg.: low priority on that item.

[9] If required, then part of correct.

[10] That is, it does what it is supposed to do.

QUESTION 10: What are the relative costs involved in your project development? For example, what percentage of the costs are spent on verification? On design?

ANSWERS: (Bulleted items are individual responses by the participants.)

- Design and coding highest; verification least; integration and testing - median.
- Definition, 15%; Design, 15%; Implementation, 15-20%; T&E, 15%; Support 40%. (Including upgrade, etc, correcting design faults.)
- Cannot break out development; verification = ∿30% of design development.
- Little is spent on design or concept validation - unless it is a software driven test bed. Most of the investment is in system development, hardware build, and testing.
- Requirements, 10%; Design, 20%; Build, 10%; Verification/ Documentation, 60%.
- <50% Verification, Integration, Testing; >10% Requirements; ≈30-40% Design/Cost/Debug.

QUESTION 11: How much effort is involved in training personnel[11]? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Training Methods
73	Special courses
36	In-house seminars
9	Invited speaker seminars
45	Technical exchanges
9	Other (explain)
	• On the job training

[11] Not enough.

QUESTION 12: What are the functions you perform (and in what order) to continuously improve your project development process[12]? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses		Responses by Importance[13]				
45	Periodic reviews	3	2	1	1	2
45	Meetings	1	1	2	3	1
36	Memo series[14]	4	4		4	4
27	Systematic approval mechanisms	2	3	3	2	3

Answers: Other (explain)

- Show personal interest in personnel by being visible in their area, asking pertinent but not antagonistic quesitons, showing interest.

QUESTION 13: If you had to choose an absolute goal as a manager, what would it be?

ANSWERS: (Bulleted items are individual responses by participants.)

- Satisfy the user!
- Abolish questionnaires that use essay questions.
- On time, under budget, totally acceptable product to the user.
- To create an environment in which personnel can work productively.
- Delivery on time, within cost, of a workable and supportable system.
- Be rich!
- Have reasonable latitude in design decisions.
- To win customer approval, corporate favor, an expanding business base.
- Make sure everyone is productive all the time, using available tools and techniques through training and on-the-job execution.

---

[12]Should change "development" to "management".

[13]Some respondents prioritized their aswers as shown here.

[14]Decision logs; not automated.

CHARACTERISTICS OF A MANAGER

QUESTION 1: Rate the following on a scale of 1 to 18 (where 18[1] is the highest priorty) the importance of personality traits for a manager.

ANSWERS:

Personality Traits	Responses by Importance									
Sense of humor	0	4	16	9	12		13	4	*A*	5
Humble	3	5	17	6	14		17	3	*B*	3
Confident[2]	17	10	14	7	18	✓	10	13	*B*	10
Flexible to change	15	9	6	13	18	✓	3	16		15
Fair[3]	8	14	3	5	18	✓	7	11	*C*	7
Compassionate[4]	4	6	12	3	18		19	6	*B*	12
Understands people	9	13	2	10	18	✓	5	18		11
Technically competent	18	12	9	14	12	✓	6	12	*D*	4
Awareness[5]	10	8	8	18	14		4	8		6
Possitive attitude	14	16	5	8	14	✓	1	17		14
Decisive[6]	11	17	7	17	16	✓	12	10	*B*	17
Delegation of responsibility	13	11	4	16	18	✓	11	15	✓	16
Aloof	5	2	18	1	1		16	2	*E*	
Integrity	16	18	1	15	18	✓	2	14		18
Discrete	6	7	13	2	17		19	5	*B*	9
Loyalty[7] [8]	7	15	11	11	18		14	7	*A*	13
Pragmatic	12	3	10	12	18	✓	20	9	*F*	8
Other										
• Systematic				4						
• Managerically competent				14						
• Focus[9]					17					
• Organizational ability							19			
• Written communication skills							9			
• Oral communication skills							12			

[1]Backwards.
[2]Too much can hurt.
[3]Essential.
[4]Desirable, not essential.
[5](or lack thereof) of your own capabilities.
[6]Nice, if got others.
[7]Up and down both.
[8]With objectivity.
[9]At high level problems - not details.

KEY
A - Top Middle
B - Low
C - High
D - Middle
E - Last
F - Very High
✓ - Neccessary. but not prioritized.

QUESTION 2: What incentives do you provide for people? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Incentives
55	Emphasis on merit increases
73	Compliments for good work
9	Optional tools to avoid drudgery
82	Encouragement to be creative
55	Promotions
36	Interpretation of organizational position
Aswers:	Other (explain)

- New titles?!

QUESTION 3: What enforcements do you execute with respect to people? (Respondents were asked to check those that applied.)

ANSWERS: % of Responses	Enforcements
18	Set working hours
36	Set techniques or tools
36	Position in personnel hierarchy is maintained
55	Working relationships are maintained[10]
Answers:	Other (explain)

- Flexible time
- Deliverable goals
- Mission accomplishment, responsible for actions, resource accountability
- No set working hours
- Negotiate work, objectives and review

QUESTION 4: What tradeoffs do you make with respect to reliability and cost effectiveness?

ANSWERS: (The bulleted items on the following page are individual responses by the participants.)

[10]But flexible.

CHARACTERISTICS OF A MANAGER: Question 4 responses.

- In many military (tactical) systems, reliability is dictated and the tradeoff is not an option.
- Reliability can be traded off for cost only after availability to accomplish the military mission is satisfactory.
- Under design to cost, a certain threshold of performance and reliability, beyond that is frosting subject to priority deletion by cost factors.
- Systems analysis at start of project.
- Reliability 70%.

QUESTION 5: Select a system.

(Ninety-one percent of respondents did not respond to this question.)

a) List all of the systems that exist within the environment of that target system and rate them.

	SYSTEM	HELPFUL	HINDRENCE	OBSOLETE	NECESSITY
TOS:	HOL	✓			✓
	Simulator				✓
	Assembly		✓	✓	

b) What new systems would be desirable to have exist within the environment of that target system?

Answer: Off-the-shelf "standardization" products, (i.e., computers, peripherals, support software, GFE products).

QUESTION 6a: How many people interface with you above your level?

b: On the same level?

c: Working for you?

d: How many people work for you in total.

e: Do some of these interfaces conflict with each other.

f: Are some non-existent?

[Author's note: Responses to this question kept proprietary.]

QUESTION 7a: How closely does your personnel structure correspond to your project and project development structures? (Respondents were asked to check one that most applied.)

ANSWERS:

% of Responses	
0	Not related
64	Related
18	Hard to tell

b: Explain.

ANSWERS: (Bulleted items are the individual responses of participants.)

- Separation of analysts, programmers, and tied together by technical support as independent evaluator.
- Usually a project will be staffed by drawing resources from various elements within the center. Matrix management.
- Contractors segmented into phases and specific capabilities for system development in the whole.
- We are a matrix organization so my organization has several people working on different projects.
- Separately into functional groupings.

QUESTION 8: Do you have a means of integrating the various people functions as well as reviewing the performance of your people that you consider to be successful?

ANSWERS: % of Responses	Responses
36	Yes
9	Reasonably so
9	Not effectively
9	No

Explanations: (Bulleted items are individual responses by participants.)

- Yes, establish teams of various resources to accomplish given task based on areas of expertise.
- Yes, we can shift people into various roles for the benefit of the individual and the job to be done.
- Cross-fertilize and cross track. Back up capability.

QUESTION 9: Do you have checklists for yourself and your people. If so, what are they?

ANSWERS: % of Responses	Response
18	Yes
36	No
9	No response

- A very formal process of documenting performance on an appraisal form. Must describe technical competence, verbal/written communications ability, cooperation, etc. Must recommend further training, ability to perform and level of performance on assigned job, etc.
- Standard Government issue, performance appraised and job description tools.
- Acquisition procedures.

APPENDIX I: ACQUISITION QUESTIONNAIRE

PART 2*

*Part 2 responses have not been summarized here. A follow-on to this effort would make use of the responses to Part 2.

*FOUNDATION FOR FORMAL METHODS

1. Suppose you were asked to list a set of general principles to adhere to in defining a system. What would they be? _____

2. Could rules be derived from these principles? _____

3. How would you train others with respect to your principles and rules?

4. Can you think of a communication problem where an object is confused with its name? What is it? _____

*FOUNDATION FOR FORMAL METHODS (continued)

5. Can you think of a communication problem where an object was confused with the definition, implementation, description or execution of that object? What was it? _____

_____

_____

_____

_____

6. What ways do you abstract with respect to system definitions for people, hardware, etc? Are they standard? _____

_____

_____

_____

_____

7. How do you share common tools, common modules, common expressions, etc.? Check those that apply.

__ Library
__ Word of mouth
__ Common manager
__ Other (explain)_____

_____

_____

8. What objects do you describe when defining a system? What characteristics about these objects are important to describe? _____

_____

_____

_____

_____

9. How much emphasis is placed on powerful but simple notation techniques?

_____

_____

_____

_____

*FOUNDATION FOR FORMAL METHODS (continued)

10. Do definitions include control considerations? Check those that apply.

__ Access rights
__ Functions to be invoked
__ Data flow
__ Ordering
__ Error detection and recovery
__ Other (explain)_____

_____

_____

11. Specifically, do definitions address real-time considerations (e.g., timing, priorities)? _____

_____

_____

_____

_____

12. Are you personally involved in defining requirements? What type of logic patterns do you and those you communicate with use (e.g., a decision)?

_____

_____

_____

_____

13. What operations do you use over and over again on a current project (e.g., a particular form of message processing)? _____

_____

_____

_____

_____

14. What functional processes would be beneficial by freezing them for later use or reconfiguration? _____

_____

_____

_____

*FOUNDATION FOR FORMAL METHODS (continued)

15. What data types do you commonly refer to (e.g., a particular message format)? _____

_____

_____

_____

_____

16. What systems might be used for more than one application, computer, or reconfiguration? _____

_____

_____

_____

_____

17. What types of information should be recorded for later use? Check those that apply.

__ Errors
__ Complaints
__ Wish lists
__ Successes
__ Result of tests
__ Manual procedures
__ Other (explain) _____

_____

_____

_____

*REQUIREMENTS CHARACTERISTICS

1. What is your definition of a requirement? _____

_____

_____

_____

_____

*REQUIREMENTS CHARACTERISTICS (continued)

2. When does a conceptual idea (or cloud) become a requirement? _____

3. How do you distinguish requirements from specifications? _____

4. What categories of requirements do you have in your system? How are they partitioned with respect to resources? _____

5. When do you distinguish hardware from software from users functions? How do you distinguish these functions? _____

6. How dependent are applications requirements from target machines and target systems? _____

7. Do requirements address memory and timing applications? How are these estimates made? _____

*REQUIREMENTS CHARACTERISTICS (continued)

8. For software requirements, how are applications modules, systems software modules, and system support tools differentiated? _____

_____

_____

_____

_____

9. Do requirements address error detection and recovery? Do they consider redundancy back-up systems? How? _____

_____

_____

_____

_____

10. Is there an attempt to save memory and time resource during the requirements phase?_____

_____

_____

_____

_____

11. What type of information is included in a requirement? Check those that apply.

__ Functions
__ Structures
__ Data types
__ Other_____

_____

_____

12. What level of detail do you use to describe requirements?

__ English
__ Flowcharts
__ Requirements language
__ Block diagrams
__ Other (specify)_____

_____ _____

*REQUIREMENTS CHARACTERISTICS (continued)

13. How are interfaces to other requirements defined? _____

14. How are inputs and outputs characterized? _____

15. How are constraints characterized? _____

16. Are both nominal and off-nominal conditions for the operation of a system defined? _____

17. Are restrictions imposed on and imposed by operational techniques, hardware systems, software systems, checklists, users, etc? _____

18. What type of protection is incorporated in the system from human errors?

19. Are effects of each module with respect to simulators, users, etc. recorded? _____

*REQUIREMENTS CHARACTERISTICS (continued)

20. Are hardware features or system software features available which aid in the definition of requirements? _____

21. What differences exist between the host environment and target environment that affect the requirements? _____

22. How are your requirements originally created (e.g., the existence of a threat)? How many groups of people and/or organizations are involved in defining the requirements? _____

LIFE-CYCLE ASPECTS OF SYSTEM DEVELOPMENT

1. Do the same people implement the requirements that design the requirements? _____

2. Is there an official review process for the integration of requirements?

3. Can you track the history of a requirements change? _____

LIFE-CYCLE ASPECTS OF SYSTEM DEVELOPMENT (continued)

4. Can you track the history of an anomaly? _____

_____

_____

5. What happens to an error when it is too late to fix the requirements?

_____

_____

_____

6. What impacts are considered for each requirement or requirement change? Check those that apply.

__ Support tool change (e.g., simulator)
__ Personnel training change
__ Mission change
__ Job security
__ Schedules
__ Other (specify)_____

_____

_____

7. Are support systems changed to correspond with relevant system changes?

_____

_____

_____

8. Are requirements changes or errors traced for second and third order effects in a system? _____

_____

9. What types of development plans and milestones are made for requirements definitions?

__ Customer review
__ In-house reviews
__ Acquisition checkpoints
__ Other (specify)_____

_____

LIFE-CYCLE ASPECTS OF SYSTEM DEVELOPMENT (continued)

10. Are standard forms used? Check those that apply.

    __ New requirements
    __ Changes to requirements
    __ Errors reported
    __ Others

*11. What is the approval hierarchy and approval focal point for new requirements, changes to requirements, and anomaly fixes? Is there a central clearing house? Are there official sign-offs? Is there a numbering system? When does it occur in the life cycle. _____

*12. What are the various technical responsibilities and technical disciplines involved in the requirements phase? _____

13. Is there a method for not letting requirements "slip through the cracks?"

14. Are there methods for tracking changes, anomalies, fixes throughout the system, support systems, and users of the system? _____

LIFE-CYCLE ASPECTS OF SYSTEM DEVELOPMENT (continued)

15. Are there methods for initiating action items and the closing of these action items? _____

16. Are there methods of introducing improvements into the requirements phase as a result of previous problems found? _____

17. Is there any attempt to make common use of modules in various parts of a system? Are common modules used in systems and their respective support systems, or is this considered a problem of generic errors? _____

18. Are specifications checked to see if they meet the requirements? _____

19. How far through the development process is a requirement monitored for consistency? _____

20. Is there independent verification of requirements? _____

LIFE CYCLE ASPECTS OF SYSTEM DEVELOPMENT (continued)

21. Are there methods for publishing vital and fast turn-around information to users of a system? _____

_____

_____

_____

22. What is the education and the background of the engineers and management involved in the design of the requirements? Give percentages.

___% of Phds

___% of Masters

___% of Bachelors

___% of 15 - 20 years experience

___% of 10 - 15 years experience

___% of 5 - 10 years experience

23. What background mix is most advantageous?_____.
Explain _____

_____

_____

_____

*TOOLS AND TECHNIQUES†

1. Software tools available in/on the data processing system(s) used by your installation: Check those that apply.

	Avail. or known at installation	Most programmers and Prog. Supervisors have been trained in/on	Written organizational guidance on use exists	Used by most programmers	New software development required to use	Use of tool is monitored/evaluated
Automated documentation						
Source text manipulation						
Program optimization						
Aids built into compilers						
Special programming languages/compilers						
Preprocessors						
Program performance evaluation						
Requirements/specification languages						
Others (specify)						

_____

_____

†Excerpts taken from Computer Software Review: The Use of Tools and Techniques, United States General Accounting Office.

TOOLS AND TECHNIQUES (continued)

2. Software techniques in your installation: Check all items that are true in the matrix below.

	Avail. or known at installation	Most programmers and Prog. Supervisors have been trained in/on	Written organizational guidance on use exists	Used by most programmers	New software development required to use	Use of technique is monitored/evaluated
Code arrangement						
Descriptive documentation						
Performance documentation						
Embedded documentation						
Programming practices/ standards						
Re-use of already written code						
Quality assurance organization/management						
Requirements/specification standards						
Programming organization/ management						
Others(s) (specify)						

_____

_____

_____

TOOLS AND TECHNIQUES (continued)

3. Does your installation have formal written rules or standards for: Check those that apply.

__ Acquisition of software tools?
__ Development of software tools?
__ Use of software tools?
__ Acquisition of software techniques?
__ Development of software techniques?
__ Use of software techniques?
__ Evaluation of effectiveness of tools after use?
__ Evaluation of effectiveness of techniques after use?

4. Please complete the sentences below by checking one of the listed options. Check only one for each statement.

	Tools only	Techniques only	Both	Neither	Don't know
Cost/benefit studies are required for the acquisition/development of					
Before general adoption, pilot projects are required to evaluate the benefits of					

TOOLS AND TECHNIQUES (continued)

5. Please indicate the benefits, if any, to your installation of each of the listed software tools. Check those that apply.

	1 Not in installation	2 No benefit	3 Probably fewer errors, better quality (can't quantify)	4 Probable lower maintenance effort (can't quantify)	5 Quantifiable benefit (dollars, staff time)	6 Documented before/after example exists
Automated documentation						
Source text manipulation						
Program optimization						
Aids built into compilers						
Special programming languages/compilers						
Preprocessors						
Program performance evaluation						
Requirements/specification language						
Other(s) (specify)						

_____

_____

_____

TOOLS AND TECHNIQUES (continued)

6. Please indicate the benefits, if any, to your installation of each of the listed software techniques. Check those that apply.

	1 Not in installation	2 No benefit	3 Probably fewer errors, better quality (cannot quantify)	4 Probable lower maintenance effort (cannot quantify)	5 Quantifiable benefit (dollars, staff time) documented before/after comparison	6
Code arrangement						
Descriptive documentation						
Performance documentation						
Embedded documentation						
Programming practices standards						
Re-use of already written code						
Quality assuarance organization/management						
Requirements/specification standards						
Programming organization/management						
Other(s) (specify)						

_____

_____

_____

TOOLS AND TECHNIQUES (continued)

7. Please estimate the benefits for the new software tools and techniques your organization has adopted in the last four (4) years. Check one column for each item.

	0-10%	11-20%	21-30%	31-40%	41-50%	Over 50%
	1	2	3	4	5	6
Improvement in programmer productivity in software development (Reduced development cost)	☐	☐	☐	☐	☐	☐
Improvement in programmer productivity in software maintenance (reduced maintenance cost)	☐	☐	☐	☐	☐	☐
Reduced overall development cost	☐	☐	☐	☐	☐	☐
Reduced overall maintenance cost	☐	☐	☐	☐	☐	☐
Other (Specify) _____	☐	☐	☐	☐	☐	☐

8. If benefits have been obtained from the use of either tools or techniques, have they been documented in a formal summary report?

   __ 1. Yes
   __ 2. No

9. If a formal report exists (yes to #8), will your installation share it with us?

   __ 1. Yes
   __ 2. No

TOOLS AND TECHNIQUES (continued)

10. Tools portability: To the best of your knowledge, which of the items in the matrix below apply to the tools now present in your installation? Check those that apply.

	Unique to your installation	Tools/benefits have been shared with other installations	In common use	Available in only one brand of hardware	Known to be available in 3 or more brands of hardware
Automated documentation					
Source text manipulation					
Program optimization					
Aids built into compilers					
Special programming languages/compilers					
Preprocessors					
Program performance evaluation					
Req'ts/Spec. language					
Other(s) (specify)					

_____

_____

_____

TOOLS AND TECHNIQUES (continued)

11. Techniques portability: Please check the items in the matrix below that apply to the techniques now used at your installation.

	Unique to your installation	Techniques/benefits have been shared with other installations	In common use
Code arrangement			
Descriptive documentation			
Performing documentation			
Embedded documentation			
Programming practices standards			
Re-use of already written code			
Quality assurance organization management			
Requirements/ specification standards			
Programming organization/ management			
Other(s) (specify)			

_____

_____

_____

TOOLS AND TECHNIQUES (continued)

12. What is your installation's preferred source of software tools? Check one.

__ We use hardware-vendor-supplied tools only.

__ We prefer to buy tools (over and above hardward-vendor-supplied) from external sources off the self.

__ We prefer to have software tools (over and above hardware-vendor-supplied) custom-built by external sources.

__ We use or would use any of the above-named sources depending on the situation.

__ Other (specify)_____

_____

_____

_____

*ANALYSIS

1. What are the major categories of anomalies discovered in requirements definition? Estimate relative importance of each of the following categories (in addition, add categories missing here):

__ Incorrect requirements

__ Incorrect change made to requirement

__ Deficient

__ Inconsistent

__ Not interpreted correctly

__ Clerical translation

__ Unclear

__ Did not plan for system constraints

__ Omission

__ Poor philosophy

__ Not flexible

__ Other (specify)_____

_____

_____

ANALYSIS (continued)

2. How is an error defined? _____

_____

_____

a) What is a software error? _____

_____

_____

b) When is an error an error from a practical point of view? _____

_____

_____

c) If a computer program "doesn't work" because of a wrong specification, is the computer program in error? _____

_____

_____

d) If an error is officially known before flight, but an official decision is made not to fix it, is it an error if it occurs during flight? _____

_____

_____

e) If the specification is incorrect, but the computer program is not, who is in error? _____

_____

_____

f) If there is an error in the input to the computer program, is the computer program in error if problems result from that input? ____

_____

_____

g) Who is responsible for error detection and recovery, if there is such a thing? _____

_____

_____

ANALYSIS (continued)

h) If an error is detected and recovered from, is there an error? ____

_____

_____

i) If two errors cancel each other, is there an error? _____

_____

_____

j) Is "better the enemy of good" in providing for protection against errors? _____

_____

_____

k) When more than one specification exists, and they differ, which one is in error? _____

_____

_____

l) If there are many errors and they have one root source, how many errors are recorded?

3. How are most anomalies found? Check those that apply. List percentages

__ Simulation ___%

__ Eyeball ___%

__ Independent verification and validation ___%

__ Dynamic execution ___%

__ Other (specify) _____

_____

_____

_____

ANALYSIS (continuation)

4. How are problems categorized with respect to the user? Check those that apply.

__ Catastrophic
__ Worrisome
__ Annoying
__ "Funny little things"
__ Borderline
__ Nonexistent
__ Known before occurrence, but forgotten
__ Known before occurrence, but not all ramifications are known
__ Other (specify) _____

_____
_____
_____

5. Which types of problems are not fixed? _____

_____
_____
_____

6. Who finds the errors? Check those that apply.

__ Individual module engineers
__ Special debugging engineers
__ Management
__ Systems engineers
__ Other (specify) _____

_____
_____

7. How many errors are interface errors? _____

_____
_____
_____
_____

ANALYSIS (continued)

9. What happens if requirements are known to use up too many system resources? Check those that apply.

__ Software compromised
__ New processing capability added to system
__ Requirements deleted
__ Performance demands minimized
__ Systems software changed
__ Software tools (e.g., compiler)
__ Different tools used (e.g., language)
__ Software converted to microcode
__ Recode implementation of previous requirements
__ Restrictive requirements
__ Other (explain)_____

_____

_____

10. In what way have software shortcomings (e.g., an obsolete software tool) influenced requirements and changes to requirements? Include resources (time and memory) and hardware architecture considerations.

_____

_____

_____

_____

11. What back-up capabilities exist in case of generic system errors? ____

_____

_____

_____

_____

12. What is done when a system and its support system disagree? _____

_____

_____

_____

_____

ANALYSIS (continued)

13. How often do requirements change? Check the one which most applies.

- __ Very often
- __ Often
- __ Not often
- __ Never

14. Have requirements been known to be wrong due to incorrect rumors or assumptions? _____

_____

_____

_____

15. How is proliferation of requirements avoided? _____

_____

_____

_____

16. In what ways has system software affected requirements? (For example, synchronous environments can restrict the flexibility of requirements.)

_____

_____

_____

17. What types of efficiency problems are major cost drivers? Check those that apply.

- __ Restrictions forced by lack of computer time and computer memory
- __ System development costs
- __ Costs of changing requirements
- __ Lack of proper personnel training
- __ Other (explain) _____

_____

_____

_____

COMMENTS

If you have any comments on the questionnaire or related topics, please use the space below.

APPENDIX II

QUESTIONS

FOR THE

PDSS WORKING GROUP

QUESTIONS FOR THE PDSS WORKING GROUP

1. Discuss differences between centralized PDSS by command and centralized PDSS by Battlefield function.

2. In the summary, under Section 3.5.3:

   a. How do you view the relative complexities and perturbations associated with Alternative 4 and Alternative 2?

   b. Is it also true that BFA, as well as PDSS, is a continuing development process? (That is, whereas the developing commands evolve technology, the BFA's evolve the missions). From our own experience we have found, in fact that mission requirements were much more volatile than technology requirements. For example, in the Apollo environment, the requirements for mission phases like Boost and Entry changed continuously whereas guidance, navigation and vehicle control requirements converged much sooner.

3. What other alternative implementations of the generalized software support model exist besides the 5 alternatives suggested in the executive summary? (For example, Alternative 6: decentralize by real user in the field). Alternative 7: centralize those functions which are in common with respect to command and centralize those functions which are in common with respect to BFA. Decentralize those functions which uniquely use those common functions.

4. What recommendations have been made in the area of front end requirements definition which could help to bridge the gap between the user in the field (i.e., the real user) and the system support expert. For example,

   a. If the software "code" were at the higher level of requirements definition, the user could fix it in real time in his own language.

   b. If users could speak the language of the support system, the need for additional experts could be alleviated.

   c. If system wide and hierarchical error detection and recovery techniques were incorporated into the system for user response, the need for adhoc fixes would be minimized.

   d. If all users' dialects could be compiled to a common meeting ground, users could be transferable from system to system.

5. What arguments would you provide to demonstrate the cost effectiveness of a PDSS plan?

6. How does a proponent in the plan (as designated in Figure 5 of the Summary) resolve interface inconsistencies with respect to his own system in the case where he has to deal with more than one support center?

7. What are the differences between the Army System in peacetime and the Army System in wartime? See Summary, Section 6, number 2. The reason for being of the Army System must consider this issue by the very fact of its existence. Thus, that which supported it in non-wartime would be required to work in wartime. This does not preclude, however, additional reconfiguruable measures that would be required in wartime. The point is that some systems are static and ready to go in peacetime whereas wartime is merely the dynamic operation of those "static" systems. Has the working group considered a preliminary "what" of this issue with an example of a "how"?

8. See Summary, for example, Section 4.3. Could a user training program be combined with the user performing operational testing where off-nominal use would provide stress testing that otherwise might not take place prior to battlefield use? (In this way independent verification and validation might provide a back-up to the nominal testing provided at the development command. In addition, training in use of the system is inherently provided for.)

9. Why does the concept in Figure 3 (Summary) best support concepts in Figure 1 and Figure 2 (Summary)?

    a. What is the difference between control, direct, monitor, approve, and manage (i.e., system manager's mission)?

    b. Does the system manager report to the interoperability configuration manager? (Figure 3 of Executive Summary.)

    c. Why is the generic function of the system manager different from that of the application software support manager (e.g., definition of requirements)?

    d. Clarify the field office versus system manager with respect to defining and resolving system problems.

10. Since most current PDSS systems have their own system software and computer types, and since MCF is not presently available, how would each PDSS center accommodate all of these diverse systems?

11. What is a reasonable plan for the transition from the present PDSS efforts to the new approach?

APPENDIX III

PRELIMINARIES OF

SYSTEM DEFINITION

To Solve a problem in Physics, you need to have three basic units:

- Mass
- Distance
- Time

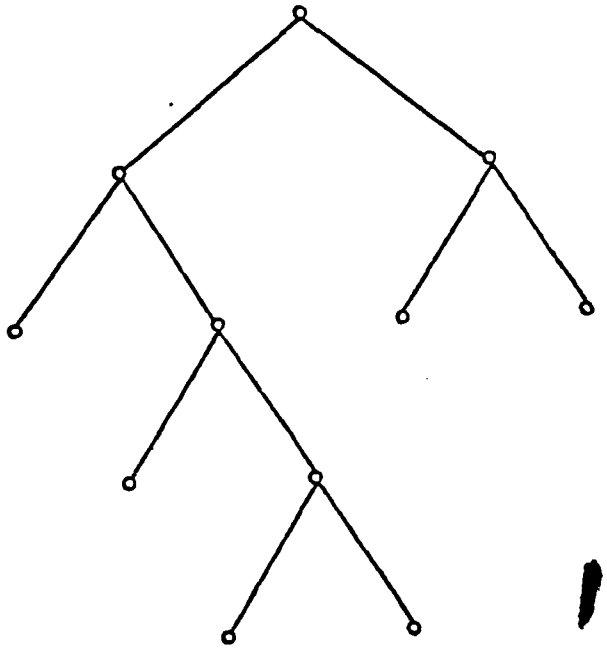
All other "units" - e.g., velocity, acceleration, momentum, energy, etc. can be expressed in terms of the three basic units.

To solve a problem in Systems, you also need to have three basic units:

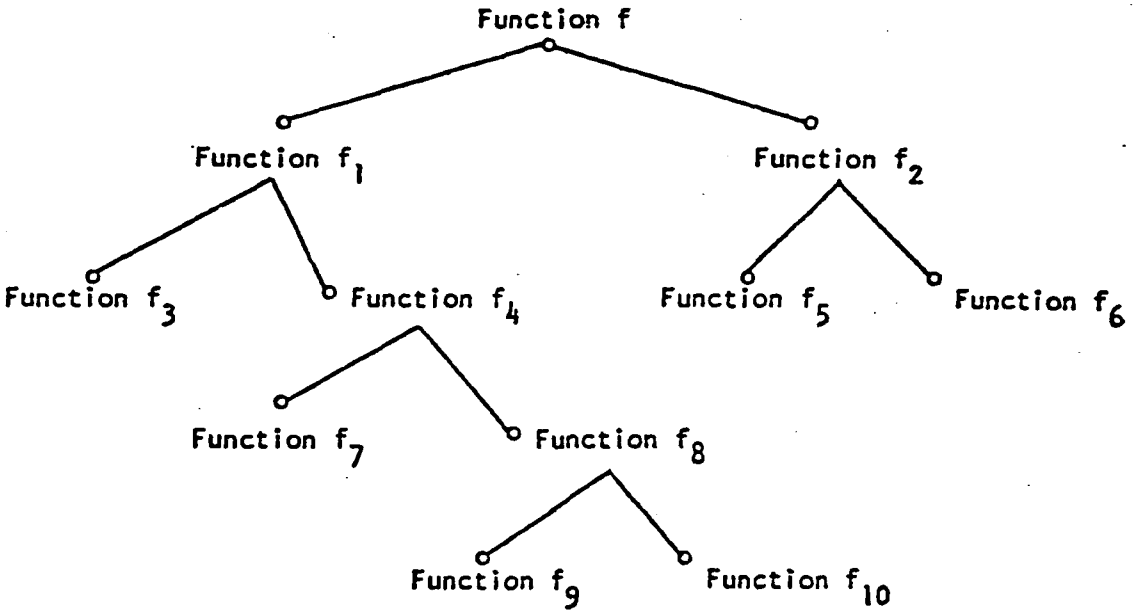
- Data Types
- Functions
- Control Structures

Other useful "units" - e.g., operations, structures, etc. can be defined in terms of the three basic ones.

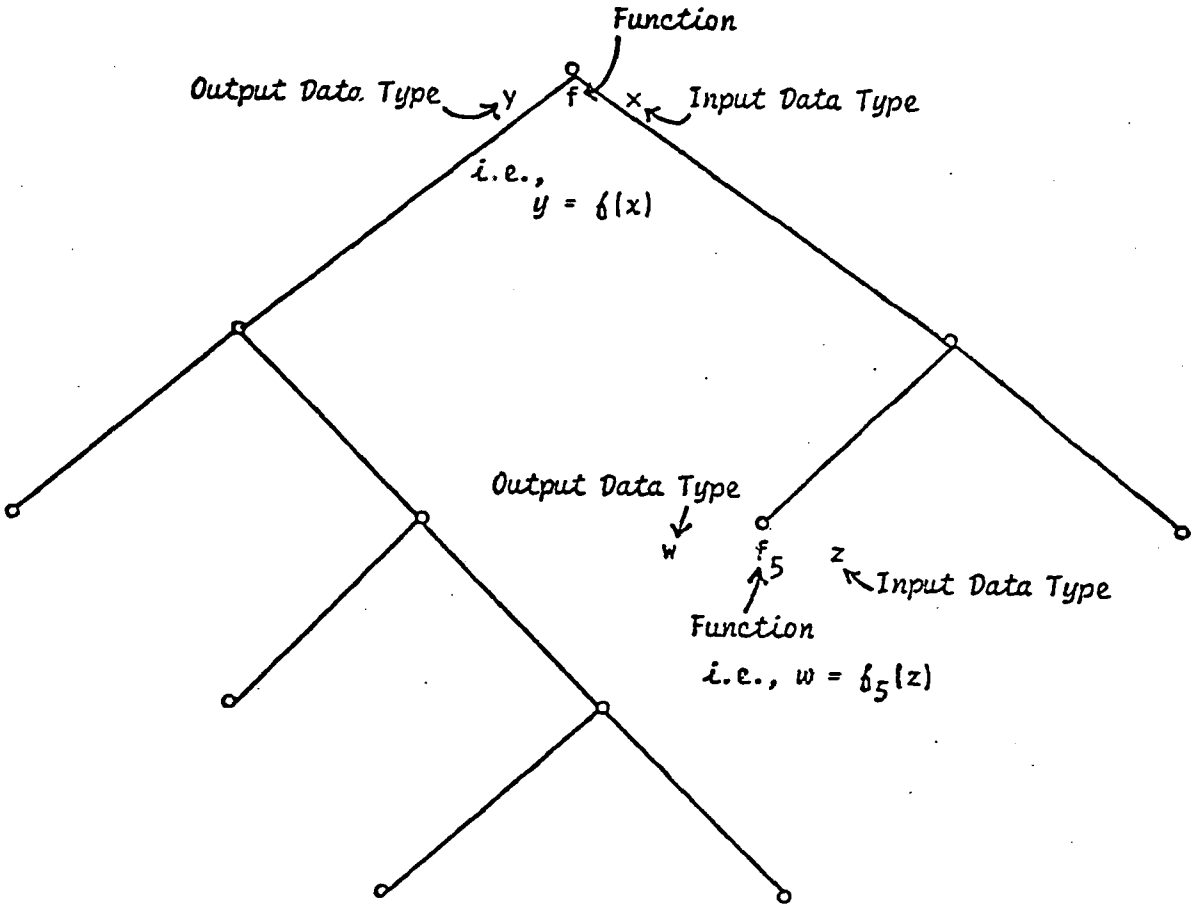
A System can be represented as a Control Map



Each Node of a control map represents a Function:

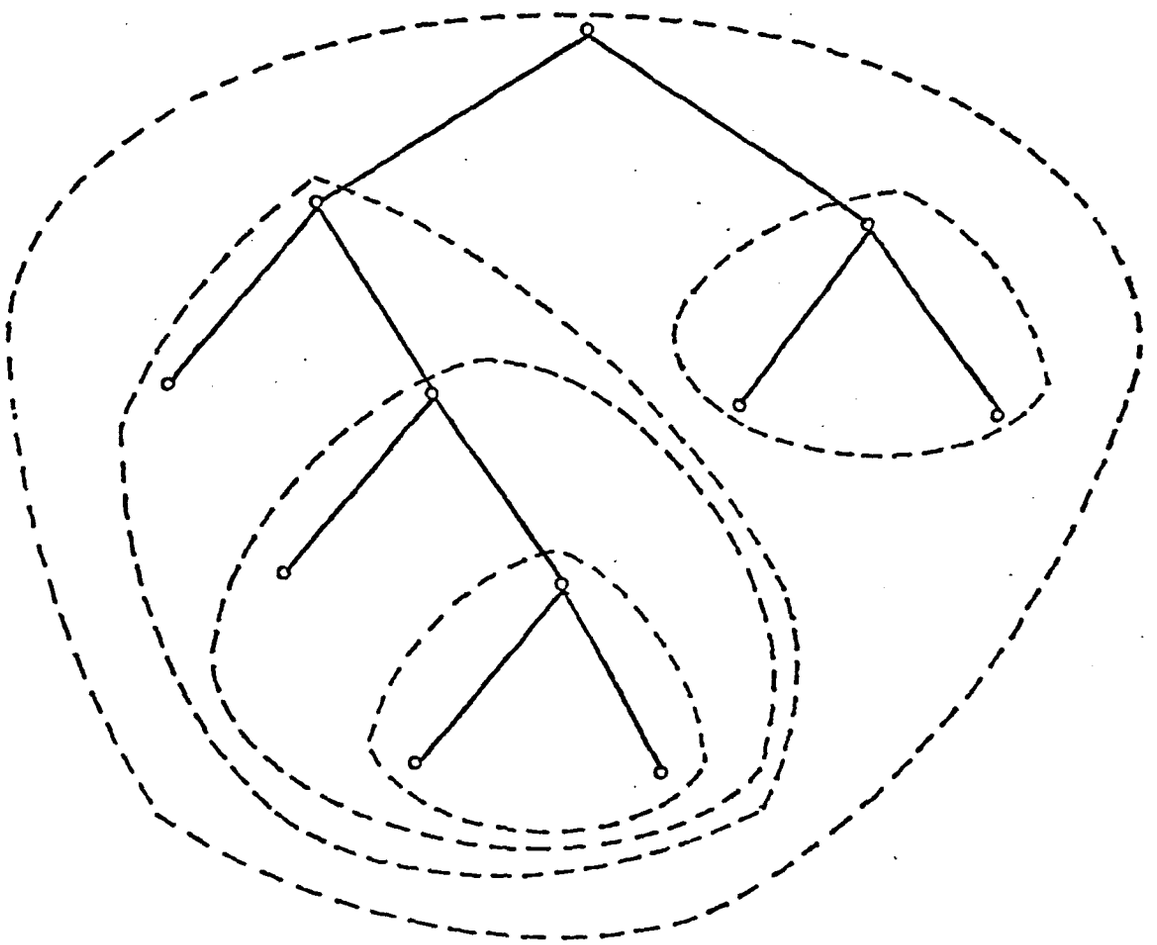


Each Function on a control map involves Data Types playing two roles - Inputs and Outputs:

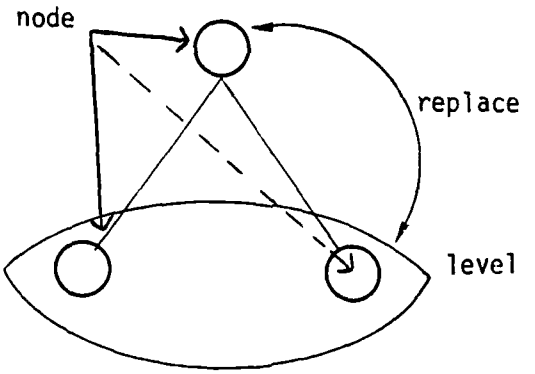


Etc.

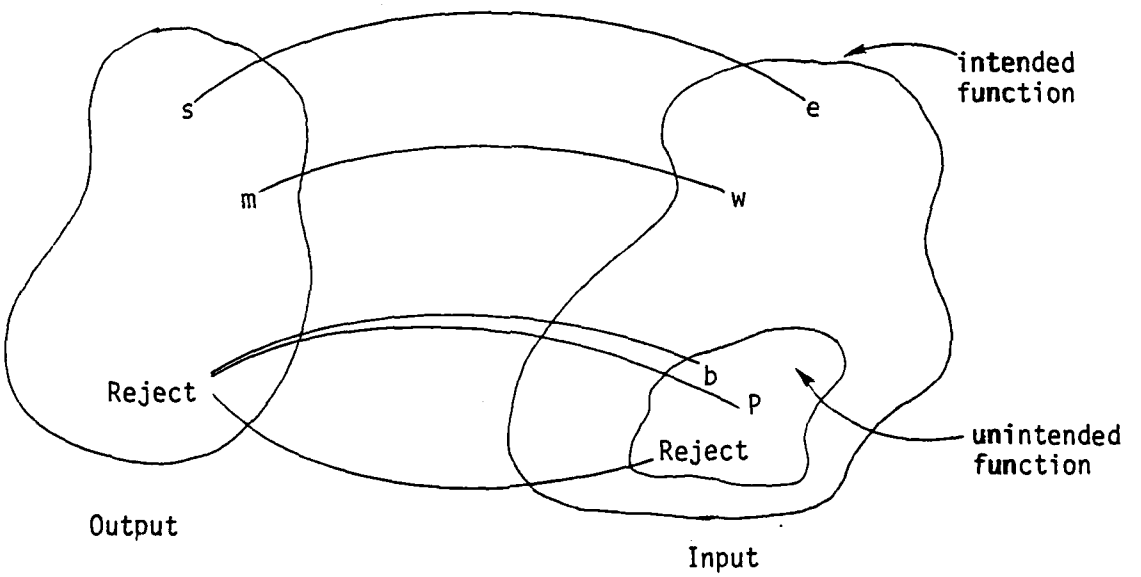
Each Decomposition on a control map represents a Control Structure:



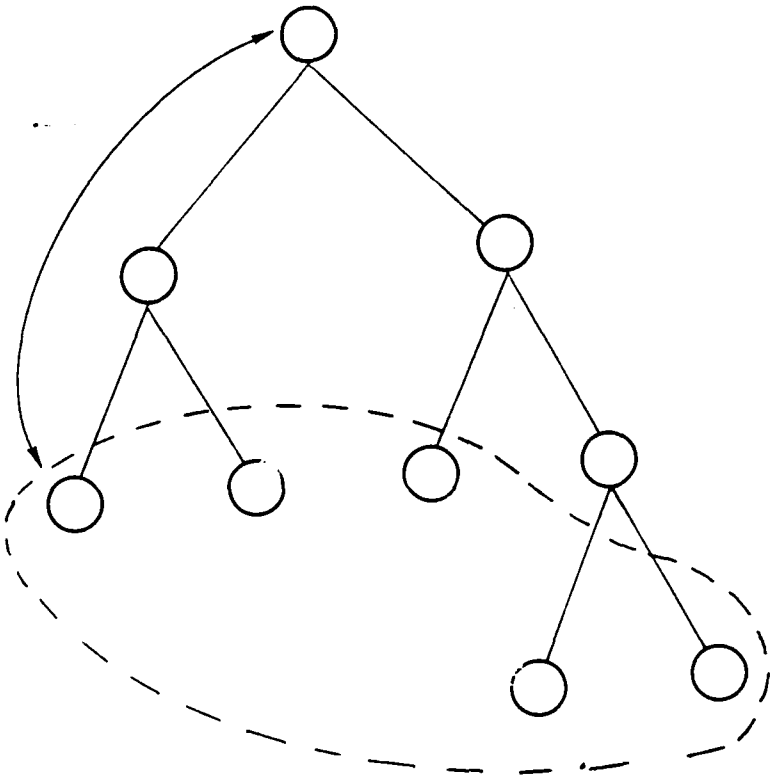
Each level of the control map completely replaces the function at the node directly above it.



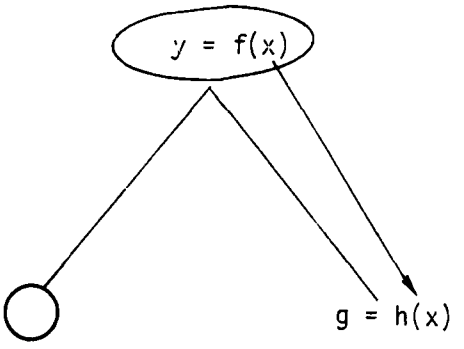
Each stopping point means a function is reached whose behavior i.e., its input/output relationship:



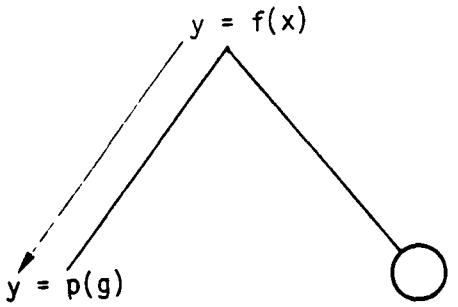
has been defined in terms of operations on a defined data type:



To talk about a value we use its name, or variable. A particular name is always associated with the same value from level to level thus input names can be traced down the control map:



And output names can be traced up the control map:



APPENDIX IV

FORMAL DEFINITIONS USED

TO DEFINE THE ARMY LIFE CYCLE

TABLE OF CONTENTS

A Control Structure For Asynchronous
Communicating Parallel Processes

When two functions communicate asynchronously an instance of one communicates with an instance of the other after one interrupts the other. In this structure definition, each instance of the particular function uses the most recent information available from its own last instance and the last completed instance of the other function in order to produce its own next result. If an instance of both functions are "ready" at the same time, the function of higher priority is that one mentioned first in the syntax statement. If there is no contention for time, both function instances may run concurrently. More than one instance of one of these functions may occur before, or during, or after an instantiation of the other instance.

The interaction, or relationship between the two functions can be seen in the control map definition in Figures A IV-1 , A IV-2 and A IV-3. In these figures, each subscripted "x" refers to a variable whose value is of data type State, and each subscripted "t" refers to a variable whose value is of type Time. The syntax for this structure of asynchronous communication is

$$x_N, x_G = N_{@Pn} \textcircled{\uparrow} G_{@Pg}(x_{N_o}, x_{G_o}, t) \,||\, \text{Stop?}$$

Here, G and N are functions of the form

$$x_{N_{i+1}} = N(x_{N_i}, x_{G_k}, t_{N_i});$$

$$x_{G_{i+1}} = G(x_{G_i}, x_{N_j}, t_{G_i})$$

where

$$x_{N_{i+1}} = \text{Ssucc}(x_{N_i}); \; x_{G_{i+1}} = \text{Ssucc}(x_{G_i})$$

$$\text{Stime}(x_{G_k}) < t_{N_i} \leq \text{Stime}(x_{G_{k+1}})$$

$$\text{Stime}(x_{N_j}) < t_{G_i} \leq \text{Stime}(x_{N_{j+1}})$$

and

$$\text{Stop?}(x_{N_i}, t_{N_i}, x_{G_j}, t_{G_j})$$

is a boolean valued function that defines the condition for terminating the execution of N and G,

t_{N_i} is the time that the ith instantiation of N

is scheduled to begin execution;

and

t_{G_i} is the time that the ith instantiation of G

is scheduled to begin execution;

t_{N_i} is calculated by function Pn

t_{G_i} is calculated by function Pg

where

$$t_{N_{i+1}} = Pn(t_{x_{N_i}}, t_{x_{G_j}}, t_{N_i})$$

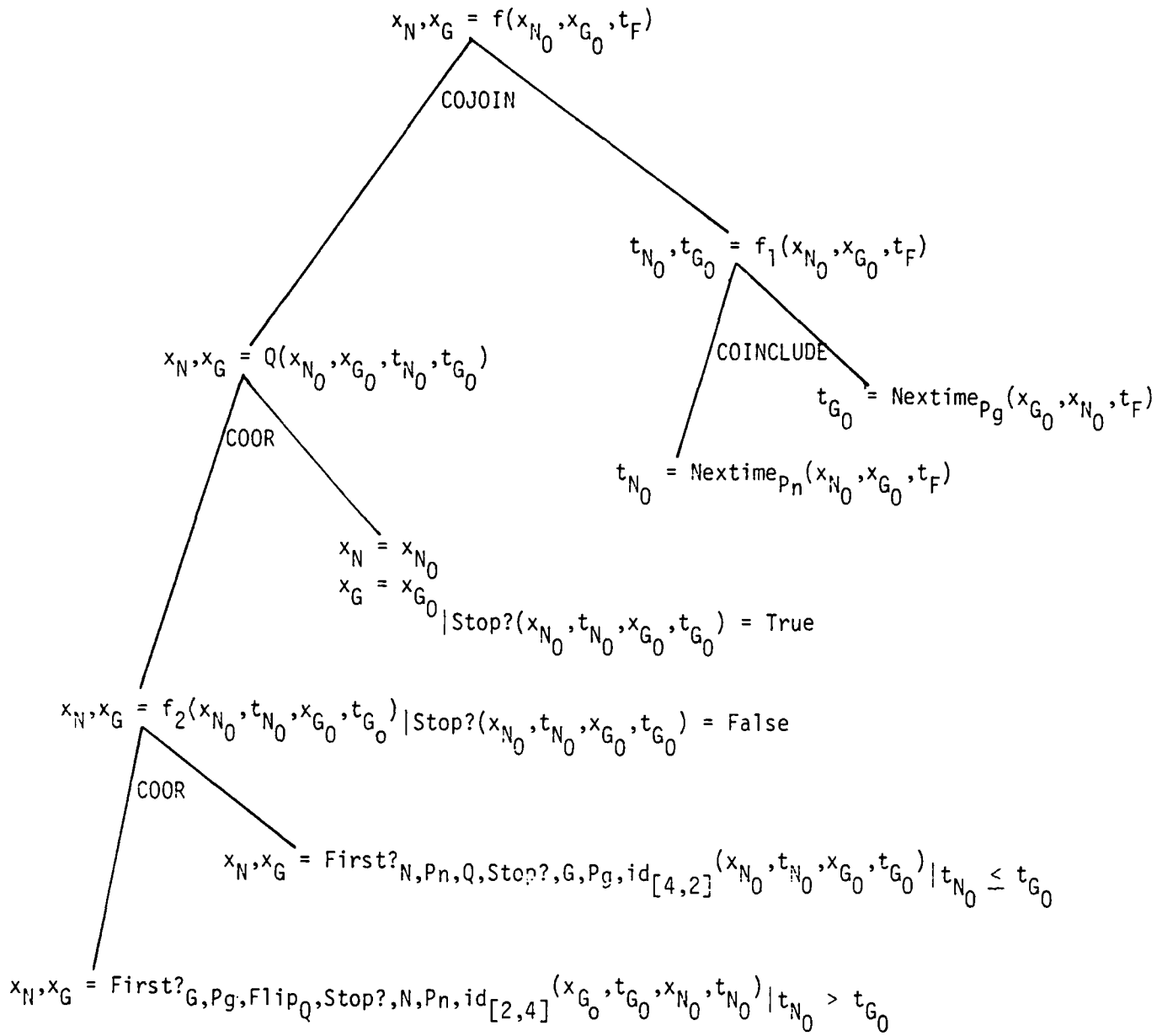
Likewise,

$$t_{G_{i+1}} = Pg(t_{x_{G_i}}, t_{x_{N_j}}, t_{G_i})$$

The last instance of G produces x_G and the last instance of N produces x_N. The control map of Figure AIV-1 assumes that the happening of two particular States and a Time indicates the execution of a particular G or N. In Figure AIV-1, the time to initiate the first instance of N (i.e., t_{N_0}) and the times to initiate the first instance of G (i.e., t_{G_0}) is produced by f_1. The offspring of f_1 use the Structure Nextime (defined in Figure AIV-4) "plugging in" functions Pn and Pg to produce t_{N_0} and t_{G_0} respectively. If the stopping criteria is met (c.f. function Stop? in Figure AIV-1) the initial input values are assigned to the final outputs x_N and x_G and struc-

ture (↑) is completed. If the stopping criteria is not met, either an instance of function N or an instance of function G will be initiated depending on tha values of t_{N_0} and t_{G_0}. The selection of which particular function to initiate first is integrated at function f_2 in Figure AIV-1. The offspring of f_2 use the structure First? (defined in Figure AIV-2) "plugging in" the appropriate functions in each case. If $t_{N_0} \leq t_{G_0}$, an instance of N is initiated first; whereas if $t_{N_0} > t_{G_0}$, an instance of G is initiated first.. The First? structure then defines the conditions under which another instance of the first selected function is initiated again, the conditions under which the second function is first initiated and the conditions under which the second function is initiated again. When G is initiated first, the ordering of the outputs x_N and x_G is specified by using the Flip structure (defined in Figure AIV-3) in the use of the First? structure. Within structure (↑), Q is initiated recursively via the use of the First? structure. This recursion synchronizes the asynchronous behavior of the instance of N and the instances of G with respect to each other.

Since there is a primitive operation on type time to advance time, but not one to reverse time, one can assume that in the " (↑) " structure definition, the output times are all greater than or equal to the input times. Thus, for example, the particular sequence of events illustrated in Figure AIV-5 could occur.



Syntax: $x_N, x_G = N_{@Pn} \textcircled{\uparrow} G_{@Pg}(x_{N_0}, x_{G_0}, t_F) || \text{Stop?}$

Fig. AIV-1 Structure $\textcircled{\uparrow}$

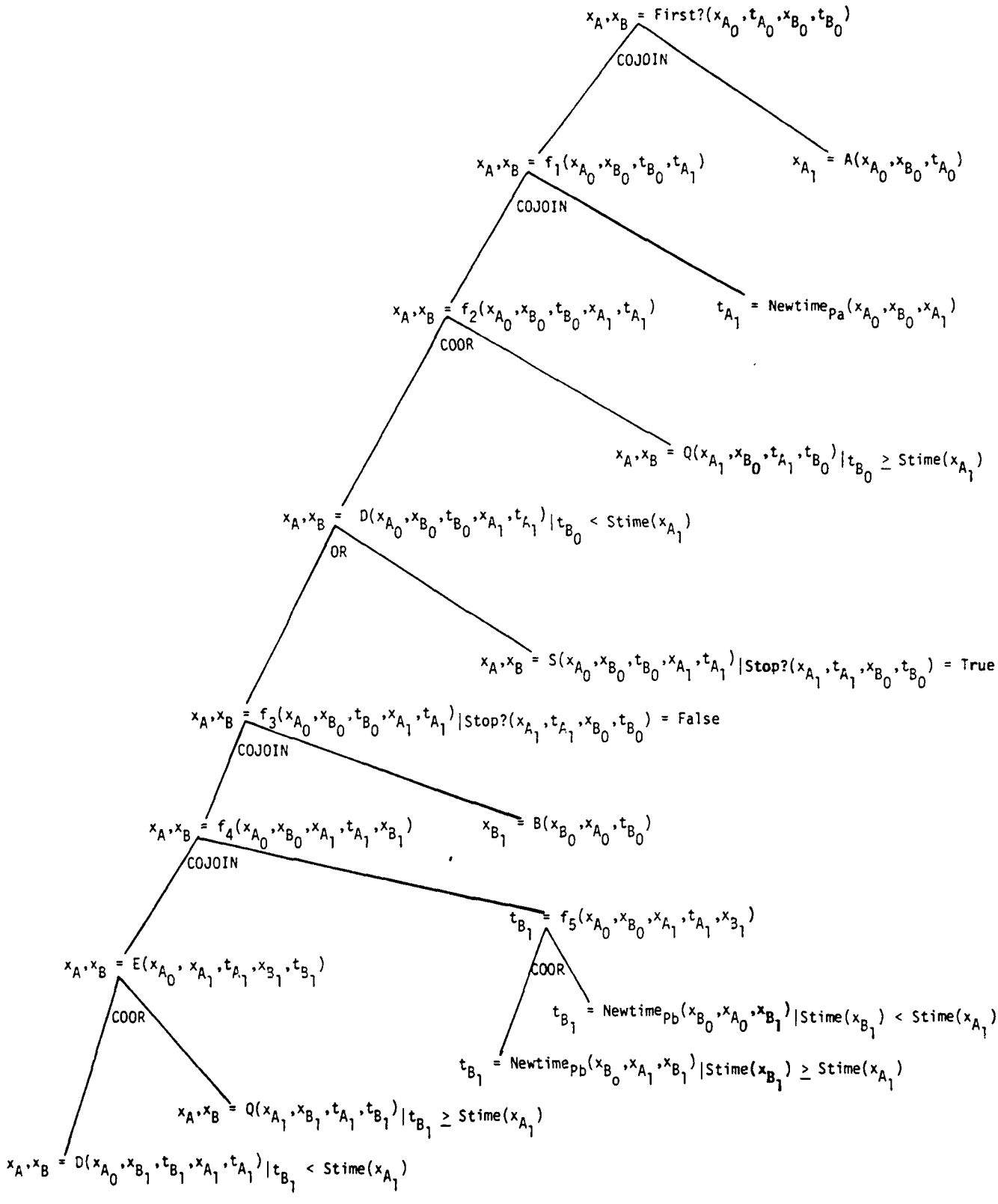
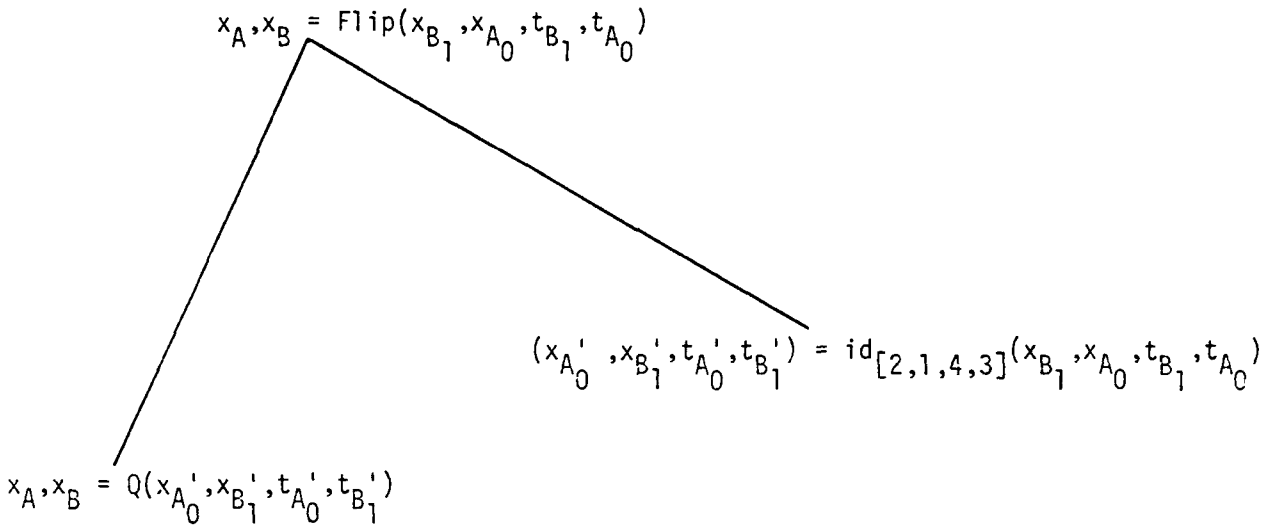
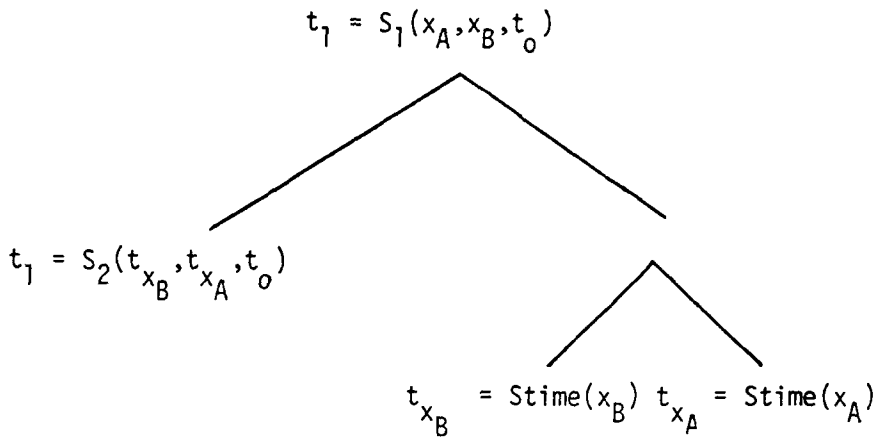


Fig. AIV-2 Structure First?

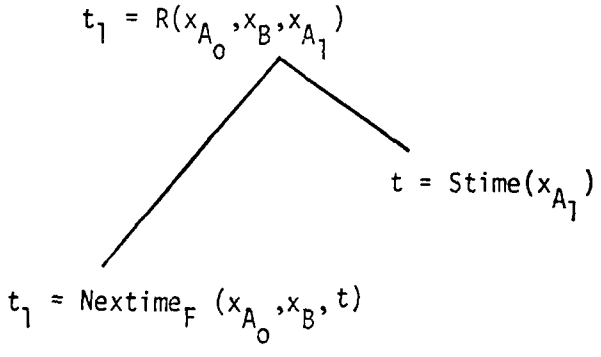


Syntax: $x_A, x_B = Flip_Q(x_{B_1}, x_{A_0}, t_{B_1}, t_{A_0})$

Fig, AIV-3 Structure Flip



Syntax: $t_1 = \text{NextTime}_{S_2}(x_A, x_B, t_o)$



Syntax: $t_1 = \text{Newtime}_F(x_{A_o}, x_B, x_{A_1})$

Fig. AIV-4 Structures Nextime and Newtime

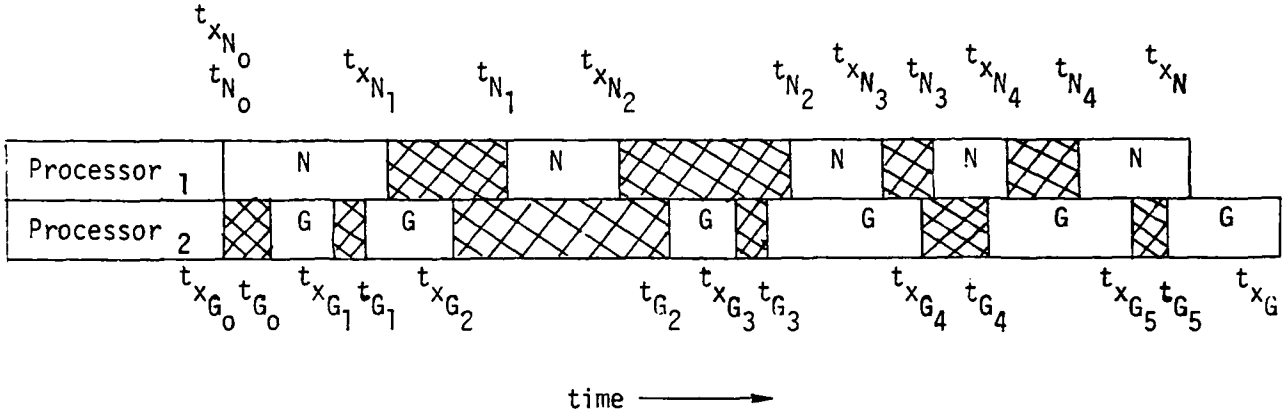


Fig. AIV-5 Example of G & N Communicating in Parallel and Asynchronously

The Failure Structure

The **failure** structure, the definition of which follows, provides for the ability to "recover" from a "detected" error. The definition uses the **cojoin, coor, join,** and **each** structures.

Structure: y = Failure(x);
where (x, g, y, $x_{[a]}$) **are of some type,**
a **is an** Ordered Set (of Naturals);

Failure: $y = f_1(x, g)$ **cojoin** $g = E(x)$;
f_1: $y = \text{Clone}_1(g)\Big|_{g=\text{Reject}}$ **coor** $y = f_2(x)\Big|_{g=\text{Reject}}$;
f_2: $y = F(x_{[a]})$ **join** $x_{[a]} = \text{id}_{[a]}(x)$;
syntax: $y = E(x)$ **failure** $y = F(x_{[a]})$;
end Failure;

DATA TYPES

TIME

DATA TYPE: TIME;

PRIMITIVE OPERATIONS:

$time_3$ = Advance($time_1$,$time_2$);

boolean = Notafter($time_1$,$time_2$);

boolean = Equal($time_1$,$time_2$);

AXIOMS:

WHERE t,t_1,t_2,t_3 ARE TIMES;
WHERE Notime IS A CONSTANT TIME;

1. Equal(t,t) = True;
2. Equal(t_1,t_2) = Equal(t_2,t_1);
3. Entails(Equal(t_1,t_2) & Equal(t_2,t_3), Equal(t_1,t_3)) = True;
4. Notafter(t,t) = True;
5. Entails(Notafter(t_1,t_2) & Notafter(t_2,t_3), Notafter(t_1,t_3)) = True;
6. Entails(Notafter(t_1,t_2) & Notafter(t_2,t_1), Equal(t_1,t_2)) = True;
7. Notafter(t_1,t_2)! Notafter(t_2,t_1) = True;
8. Advance(t,Notime) = t;
9. Advance(t_1,t_2) = Advance(t_2,t_1);
10. Advance(t_1,Advance(t_2,t_3)) = Advance(Advance(t_1,t_2),t_3);
11. Notafter(Advance(t_1,t_2),t_1) = Notafter(t_2,Notime);

END TIME;

STATE OF (T)

Data Type: State (of T);
primitive operations:
time = Stime(state);
t = Correspondent(state);
$state_2$ = Ssucc($state_1$);
Boolean = Sequals($state_1$, $state_2$);
axioms:
where (s_1, s_2) are States (of T),
time is a Time,
t is a T;
Precedes?(Stime(s_1), Stime(Ssucc s_1)) = True;
Equals(Correspondent(s_1), Correspondent(s_2))
= False ⊂ Stime(s_1) = Stime(s_2) = True;
Sequals(s_1, s_2) = Equals(Stime(s_1), Stime(s_2))
And Equals(Correspondent(s_1), Correspondent(s_2));
end State (of T);

MONEY

DATA TYPE: MONEY;

Primitive Operations:

BOOLEAN = MORETHAN (MONEY$_1$, MONEY$_2$);

/*MONEY IS ORDERED*/

MONEY$_3$ = TOTAL (MONEY$_1$, MONEY$_2$);

/*MONEY ADDS*/

From "Some Characterizations of Resources," S. Cushing (DCPA Memo in preparation).

AXIOMS:

Where M, M_1, M_2, M_3 *are* MONEYS;

MORETHAN (M,M) = FALSE;

((MORETHAN (M_1, M_2) & MORETHAN (M_2, M_3)) ⊃ MORETHAN (M_1, M_3)) = TRUE;

(($M_1 \neq M_2$) ⊃ OR (MORETHAN (M_1, M_2), MORETHAN (M_2, M_1))) = TRUE;

AND (MORETHAN (M_1, M_2), MORETHAN (M_2, M_1)) = FALSE;

/*MONEY IS ORDERED*/

TOTAL (M_1, M_2) = TOTAL (M_2, M_1);

TOTAL (TOTAL (M_1, M_2), M_3) = TOTAL (M_1, TOTAL (M_2, M_3));

/*MONEY ADDS*/

END: MONEY;

COMMODITY

DATA TYPE: COMMODITY;

Primitive Operations:

PERSON$_2$ = SELLER (COMMODITY, TIME, PERSON$_1$);

/*PERSON$_2$ IS THE SELLER OF A COMMODITY AT A TIME TO PERSON$_1$*/

PERSON$_2$ = BUYER (COMMODITY, TIME, PERSON$_1$);

/*PERSON$_2$ IS THE BUYER OF A COMMODITY AT A TIME TO PERSON$_1$*/

COMMODITY$_3$ = LOT (COMMODITY$_1$, COMMODITY$_2$);

/*A COLLECTION OF COMMODITIES IS ALSO A COMMODITY*/

BOOLEAN = APPRECIATES (COMMODITY);

/*SOME COMMODITIES APPRECIATE IN VALUE*/

BOOLEAN = DEPRECIATES (COMMODITY);

/*SOME COMMODITIES DEPRECIATE IN VALUE*/

MONEY = VALUE (COMMODITY, TIME);

/*MONEY IS THE VALUE OF A COMMODITY AT A TIME*/

AXIOMS:

Where C, C_1, C_2 *are* COMMODITIES;

Where T, T_1, T_2 *are* TIMES;

Where P *is a* PERSON;

SELLER (C,T, BUYER (C,T,P)) = P

BUYER (C,T, SELLER (C,T,P)) = P

/*BUYING AND SELLING ARE INVERSES*/

LOT (c_1, c_2) = LOT (c_2, c_1);

LOT (LOT (c_1, c_2), c_3) = LOT (c_1, LOT(c_2, c_3));

/*COMMODITIES ADD*/

MORETHAN (VALUE(LOT(c_1, c_2), T), TOTAL (VALUE (c_1,T), VALUE(c_2,T))) = FALSE;

/*THE VALUE OF A COLLECTION OF COMMODITIES IS EQUAL TO OR LESS THAN
THE TOTAL VALUE OF THE INDIVIDUAL COMMODITIES:
"ECONOMY OF SCALE"*/

```
OR (APPRECIATES(C), DEPRECIATES (C)) = TRUE;

AND (APPRECIATES (C), DEPRECIATES (C)) = FALSE;

    /*EVERY COMMODITY EITHER APPRECIATES OR DEPRECIATES BUT NOT BOTH*/


(NOTAFTER (T1, T2) ⊃ (MORETHAN (VALUE (C, T2), VALUE (C, T1)))) = Ktrue(1C) OR Kfalse(2C);

    PARTITION OF C IS

        1C| APPRECIATES (C) = TRUE;

        2C| APPRECIATES (C) = FALSE;

    /*THE VALUE OF A COMMODITY INCREASES WITH TIME IF AND ONLY IF IT IS ONE THAT
      APPRECIATES*/

END:  COMMODITY;
```

APPENDIX V

THREE PRIMITIVE

CONTROL STRUCTURES

The primitive control structures form the basis for defining other control structures in AXES. The use of AXES syntax and associated rules for the primitive control structures follow:

For composition, if $y = f_0(x)$,
f_0: $y = f_2(g)$ **join** $g = f_1(x)$;

(See Figure A1.)

1. One and only one offspring (specifically, f_1 in this example) receives access rights to the input data x from f_0.
2. One and only one offspring (specifically, f_2 in this example) has access rights to deliver the output data y for f_0.
3. All other input and output data that will be produced by offspring, controlled by f_0, will reside in *local* variables (specifically, "g" in this example). Local variable "g" provides communication between the offspring f_2 and f_1.

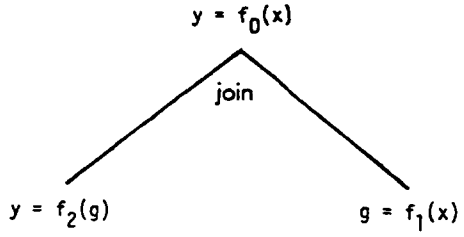


Figure A1. Composition.

4. Every offspring is specified to be invoked once and only once in each process of performing its parent's corresponding function.
5. Every local variable must exist both as an input variable for one and only one function and as an output variable for one and only one different function on the same level.

For Class partition, if $(y_1, y_2) = f_0(x_1, x_2)$,
f_0: $y_1 = f_1(x_1)$ **include** $y_2 = f_2(x_2)$;

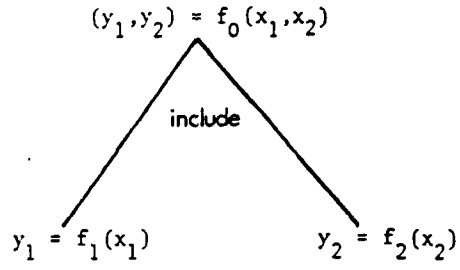
(See Figure A2.)

1. All offspring of f_0 are granted permission to receive input values taken from a partitioned variable in the set of the parent's corresponding function domain variables, such that each offspring's set of input variables collectively represents the parent's corresponding function input variables.
2. All offspring of f_0 are granted permission to produce output values for a partitioned variable in the set of the parent's corresponding function range variables, such that the sets of each offspring's output variables collectively represent the parent's corresponding function variables.
3. Each offspring is specified to be invoked per input value received for each process of performing its parent's corresponding function.
4. There is no communication between offspring.

For set partition, if $y = f_0(x)$,
f_0: $y = f_2(x)|_{property}$ **or** $y = f_1(x)|_{Pnot\ property}$;

(See Figure A3.)

Figure A2. Class partition.



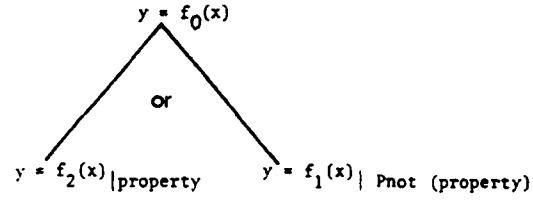
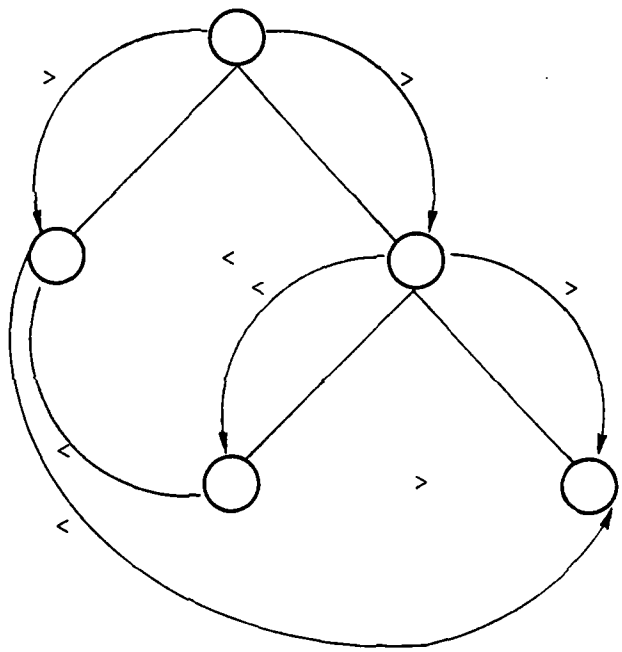


Figure A3. Set partition.

1. Every offspring of the parent at f_0 is granted permission to produce output values of "y."
2. All offspring of the parent at f_0 are granted permission to receive input values from the variable "x."
3. Only one offspring is specified to be invoked per input value received for each process of performing its parent's corresponding function.
4. The values represented by the input variables of an offspring's function comprise a proper subset of the domain of the function of the parent.
5. There is no communication between offspring.

In the above definitions x, y, y_1, y_2, x_1, x_2 are ordered sets of variables; f_0, f_1, f_2 are functions; property is of type Property (of T) [19]; and Pnot is a primitive operation on type property whose result is a property exclusive of its input argument.

Each function is always more important than the functions at the level dominated by that function, and at a particular level each function is assigned an importance with respect to each other function at that level:



The primitive control structures form the basis for defining other control structures in AXES. The use of AXES syntax and associated rules for the primitive control structures follow:

For composition, if $y = f_0(x)$,
f_0: $y = f_2(g)$ join $g = f_1(x)$:

(See Figure A1.)

1. One and only one offspring (specifically, f_1 in this example) receives access rights to the input data x from f_0.
2. One and only one offspring (specifically, f_2 in this example) has access rights to deliver the output data y for f_0.
3. All other input and output data that will be produced by offspring, controlled by f_0, will reside in *local* variables (specifically, "g" in this example). Local variable "g" provides communication between the offspring f_2 and f_1.

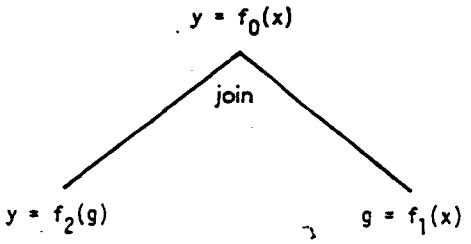


Figure A1. Composition.

4. Every offspring is specified to be invoked once and only once in each process of performing its parent's corresponding function.
5. Every local variable must exist both as an input variable for one and only one function and as an output variable for one and only one different function on the same level.

For Class partition, if $(y_1, y_2) = f_0(x_1, x_2)$,
f_0: $y_1 = f_1(x_1)$ include $y_2 = f_2(x_2)$:

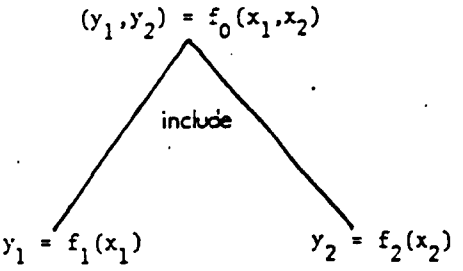
(See Figure A2.)

1. All offspring of f_0 are granted permission to receive input values taken from a partitioned variable in the set of the parent's corresponding function domain variables, such that each offspring's set of input variables collectively represents the parent's corresponding function input variables.
2. All offspring of f_0 are granted permission to produce output values for a partitioned variable in the set of the parent's corresponding function range variables, such that the sets of each offspring's output variables collectively represent the parent's corresponding function variables.
3. Each offspring is specified to be invoked per input value received for each process of performing its parent's corresponding function.
4. There is no communication between offspring.

For set partition, if $y = f_0(x)$,
f_0: $y = f_2(x)$ property or $y = f_1(x)$ Pnot property;

(See Figure A3.)

Figure A2. Class partition.



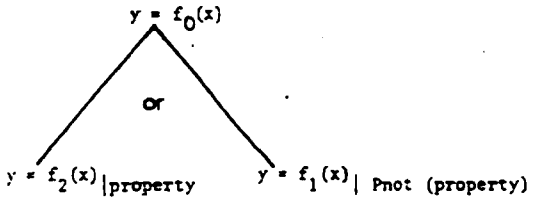


Figure A3. Set partition.

1. Every offspring of the parent at f_0 is granted permission to produce output values of "y."
2. All offspring of the parent at f_0 are granted permission to receive input values from the variable "x."
3. Only one offspring is specified to be invoked per input value received for each process of performing its parent's corresponding function.
4. The values represented by the input variables of an offspring's function comprise a proper subset of the domain of the function of the parent.
5. There is no communication between offspring.

In the above definitions x, y, y_1, y_2, x_1, x_2 are ordered sets of variables; f_0, f_1, f_2 are functions; property is of type Property (of T) [19]; and Pnot is a primitive operation on type property whose result is a property exclusive of its input argument.